

This Page Is Inserted by IFW Operations  
and is not a part of the Official Record

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS





**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning documents *will not* correct images,  
please do not report the images to the  
Image Problem Mailbox.**

# METHOD FOR RECOGNIZING CHARACTER AND DEVICE THEREFOR

**Patent number:** JP6068301  
**Publication date:** 1994-03-11  
**Inventor:** WANG SHIN-YWAN; VAEZI MEHRZAD R; SHERRICK CHRISTOPHER A  
**Applicant:** CANON INC.; CANON INF SYST INC  
**Classification:**  
 - international: G06K9/20; G06K9/32; G06K9/34  
 - european:  
**Application number:** JP19930121883 19930426  
**Priority number(s):**

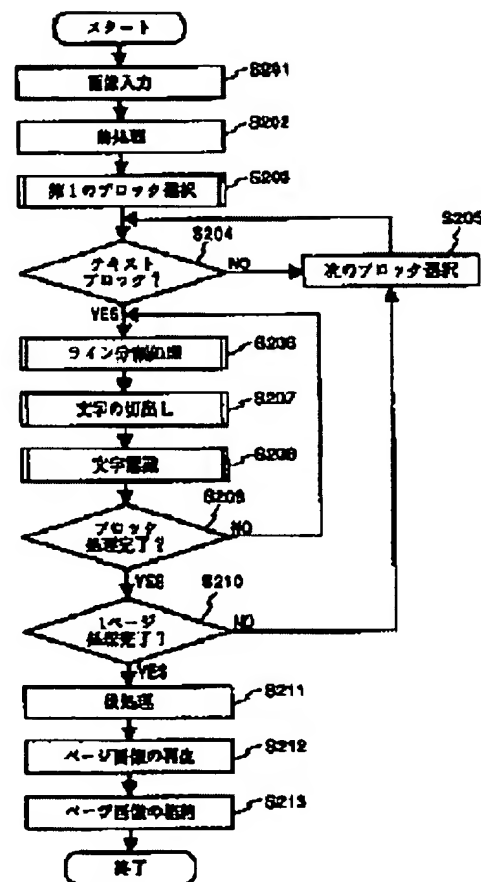
Also published as:

 EP0567344 (A2)  
 US5680479 (A1)  
 EP0567344 (A3)  
 EP0567344 (B1)

## Abstract of JP6068301

**PURPOSE:** To provide a character recognition method and device for recognizing a character on a document with a high speed and exactness, and preparing a text file.

**CONSTITUTION:** Pixel picture data of pixel units are inputted to a device (S201), the picture quality of the picture data is improved by a filter in order to improve deteriorated characters or pictures (S202), the hierarchical tree structure of the pictures is prepared (S203) so that text information/graphics information/ line information/picture information of information in a block can be identified, and each part in the picture can be reproduced in a proper sequence like a post-processing (S212), and whether or not the selected block is a text block is checked (S204). Then, a line dividing processing is operated to the text block (S206), each character in the line is segmented from another character in the line (S207), whether or not the processing to the text block and the processing to one page is ended is checked (S209 and 210), and a correction processing from an overall point of view such as context check or spelling check is operated (S211).



Data supplied from the esp@cenet database - Worldwide

T S1/5/1

1/5/1

DIALOG(R)File 347:JAPIO

(c) 2004 JPO & JAPIO. All rts. reserv.

04424401

METHOD AND DEVICE FOR RECOGNIZING CHARACTER

PUB. NO.: 06-068301 [JP 6068301 A]

PUBLISHED: March 11, 1994 (19940311)

INVENTOR(s): SHIN YAN WANGU

MEZAADOU AARU BAEZUII

KURISUTOFUAA EE SHIERITSUKU

APPLICANT(s): CANON INC [000100] (A Japanese Company or Corporation), JP  
(Japan)

CANON INF SYST INC [000000] (A Non-Japanese Company or  
Corporation), US (United States of America)

APPL. NO.: 05-121883 [JP 93121883]

FILED: April 26, 1993 (19930426)

PRIORITY: 7-873,012 [US 873012-1992], US (United States of America),  
April 24, 1992 (19920424)

INTL CLASS: [5] G06K-009/20; G06K-009/32; G06K-009/34

JAPIO CLASS: 45.3 (INFORMATION PROCESSING -- Input Output Units); 44.7  
(COMMUNICATION -- Facsimile)

JAPIO KEYWORD: R002 (LASERS); R107 (INFORMATION PROCESSING -- OCR & OMR  
Optical Readers); R108 (INFORMATION PROCESSING -- Speech  
Recognition & Synthesis); R131 (INFORMATION PROCESSING --  
Microcomputers & Microprocessers); R139 (INFORMATION  
PROCESSING -- Word Processors)

?

(19) 日本国特許庁 ( J P )

(12) 公 開 特 許 公 報 ( A )

(11) 特許出願公開番号

特開平6-68301

(43) 公開日 平成 6 年 (1994) 3 月 11 日

| (51) Int.Cl. <sup>5</sup> | 識別記号 | 庁内整理番号  | F I | 技術表示箇所 |
|---------------------------|------|---------|-----|--------|
| G 0 6 K                   | 9/20 | 3 4 0 L |     |        |
|                           | 9/32 |         |     |        |
|                           | 9/34 |         |     |        |

審査請求 未請求 請求項の数286(全 420 頁)

|              |                 |          |   |
|--------------|-----------------|----------|---|
| (21) 出願番号    | 特願平5-121883     | (71) 出願人 | 000001007<br>キヤノン株式会社<br>東京都大田区下丸子3丁目30番2号  |
| (22) 出願日     | 平成5年(1993)4月26日 | (71) 出願人 | 592208172<br>キヤノン インフォメーション システムズ インク.<br>Canon Information Systems, Inc.<br>アメリカ合衆国 カリフォルニア州<br>92626, コスタ メサ, ブルマン ストリート 3188 |
| (31) 優先権主張番号 | 8 7 3 0 1 2     | (74) 代理人 | 弁理士 大塚 康徳 (外1名)   |
| (32) 優先日     | 1992年4月24日      |          |   |
| (33) 優先権主張国  | 米国 (US)         |          |   |

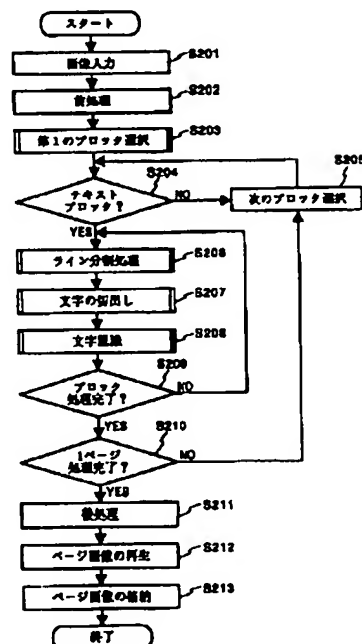
最終頁に続く

(54) 【発明の名称】 文字認識方法及び装置

(57) 【要約】 (修正有)

【目的】 高速かつ正確に文書上の文字を認識してテキストファイルを作成する文字認識方法及び装置の提供。

【構成】 画素単位の画素画像データを装置に入力し (S201)、フィルタで画像データの画質を向上して劣化した文字や画像を改善し (S202)、ブロック中情報のテキスト情報/グラフィックス情報/線画情報/画像情報などの識別と画像中の各部分を後処理 (S212) のような適切な順序で再生できるように画像の階層的な構造を生成し (S203)、選択されたブロックがテキストブロックか否かを調べ (S204)、テキストブロックに対してライン分割処理を行い (S206)、ライン中の各文字をライン中の他の文字から切り出し (S207)、テキストブロックに対する処理と1ページ分に対する処理が終了したかを調べ (S209, 210)、文脈チェックやスペルチェックなどの全体的な見地からの修正処理を行う (S211)。





1

【特許請求の範囲】

【請求項1】 画素画像データから画素ブロックを選別する方法であって、

画素データ内で連結要素の輪郭を追跡する行程と、  
輪郭が追跡された各連結要素を囲んで矩形を形成する行程と、

サイズと他の矩形への近接関係とに基づいて矩形を選択的に横方向に連結して、ラインを形成する第1の連結行程と、

サイズと他のラインへの近接関係とに基づいてラインを選択的に縦方向に連結して、ブロックを形成する第2の連結行程とを備える方法。

【請求項2】 更に、画素画像データを入力する行程を備え、

該入力行程は、画素画像データが2値画素画像データでない場合に、画素画像データを2値画素画像データに変換する行程を含む請求項1記載の方法。

【請求項3】 前記矩形形成行程では、各連結要素を囲む最小の矩形が形成される請求項1記載の方法。

【請求項4】 更に、前記矩形形成行程で形成された対応する矩形の位置に基づいて、前記輪郭追跡行程で追跡された連結要素の階層的木構造を形成する行程を備える請求項1記載の方法。

【請求項5】 更に、前記矩形形成行程で形成された矩形をテキスト要素と非テキスト要素とに分類する行程を備える請求項4記載の方法。

【請求項6】 更に、テキスト要素のブロック内の文字画像を認識する行程を備える請求項5記載の方法。

【請求項7】 前記分類行程は、要素の高さの所定のしきい値に対応して実行される請求項5記載の方法。

【請求項8】 前記分類行程は、前記矩形形成行程で形成された矩形の高さに関連する統計情報に対応して実行される請求項6記載の方法。

【請求項9】 前記第1の連結行程と第2の連結行程とは、非テキスト要素に対しては実行されない請求項6記載の方法。

【請求項10】 更に、非テキスト要素に対し白輪郭を得る行程を備える請求項6記載の方法。

【請求項11】 非テキスト要素は白輪郭の数に対応して表要素と見なされる請求項10記載の方法。

【請求項12】 更に、白輪郭の充填率を算出する行程を備える請求項10記載の方法。

【請求項13】 非テキスト要素は、充填率が高い場合には画像データと見なされない請求項12記載の方法。

【請求項14】 更に、非格子状に配置された白輪郭を再接続する行程を備える請求項12記載の方法。

【請求項15】 非テキスト要素は、再接続率が高くない場合に表と見なされる請求項14記載の方法。

【請求項16】 白輪郭は4方向で算出される請求項10記載の方法。

2

【請求項17】 連結要素の輪郭は、少なくとも8方向に追跡される請求項1記載の方法。

【請求項18】 前記輪郭追跡行程では、連結要素の外部のみで連結要素の輪郭が追跡される請求項1記載の方法。

【請求項19】 更に、画素画像データ内でギャップを検出する行程を備え、前記第1の連結行程では、ギャップが矩形を隔てている場合には矩形がラインに連結されない請求項1記載の方法。

【請求項20】 矩形の間を縦に延びるギャップに対応して段を検出する請求項19記載の方法。

【請求項21】 前記第2の連結行程は、前記第1の連結行程で連結されたテキストデータのライン間に非テキストの境界を決定する行程を含み、前記第2の連結行程は、間に非テキストの境界がある場合にラインを縦方向にブロックとして連結しない請求項1記載の方法。

【請求項22】 更に、前記輪郭追跡行程の前に画素画像データを圧縮する行程を備える請求項1記載の方法。

【請求項23】 画素画像データ内の文字を認識する方法であって、

画素画像データ内で連結要素の輪郭を追跡する行程と、  
追跡された連結要素がテキスト要素を含むか非テキスト要素を含むかを定める行程と、  
テキスト要素を横方向に選択的に連結してテキストラインを形成する行程と、  
テキストラインを選択的に縦方向に連結してテキストブロックを形成する行程とを含み、  
画素画像データから画素ブロックを選別する行程と、  
テキストブロックを画素画像データのラインに分割する行程と、

前記分割行程で分割されたラインから文字を切り出す行程と、

前記切り出行程で切り出された文字を認識する行程と、  
前記認識行程で認識された文字を、前記ブロック選別行程で確定された順に対応して格納する行程とを備える方法。

【請求項24】 更に、テキスト要素のブロック内の文字画像を認識する行程を備える請求項23記載の方法。

【請求項25】 更に、画素画像データが前処理される前処理行程を備える請求項23記載の方法。

【請求項26】 前記前処理行程は画像圧縮行程を含む請求項25記載の方法。

【請求項27】 前記前処理行程は画素画像データの画質を向上するフィルタ行程を含む請求項25記載の方法。

【請求項28】 更に、前記認識行程で認識された文字を後処理する行程を備える請求項23記載の方法。

【請求項29】 前記後処理は文脈チェックを含む請求項28記載の方法。

【請求項30】 更に、画素画像データを入力する行程を備え、

3

該入力行程は、画素画像データが2値画素画像データでない場合に、画素画像データを2値画素画像データに変換する行程を含む請求項23記載の方法。

【請求項31】 前記ブロック選別行程は、前記輪郭追跡行程で定められた連結画素に基づいて階層の木構造を形成する行程を含む請求項23記載の方法。

【請求項32】 更に、前記輪郭追跡行程で追跡された連結要素を囲む矩形を形成する行程を備える請求項31記載の方法。

【請求項33】 前記階層の木構造は、前記矩形形成行程で形成された矩形の位置に基づいて形成される請求項32記載の方法。

【請求項34】 更に、第1及び第2の連結行程に基づいて、階層の木構造を更新する行程を備える請求項31記載の方法。

【請求項35】 更に、連結要素をテキスト要素又は非テキスト要素に分類する行程を備える請求項31記載の方法。

【請求項36】 前記第1及び第2の連結行程は、テキスト要素を連結するが非テキスト要素は連結しない請求項35記載の方法。

【請求項37】 更に、非テキスト要素の内部で白輪郭を追跡する行程を備える請求項35記載の方法。

【請求項38】 更に、非テキスト要素に含まれる白輪郭の数に基づいて、非テキスト要素を表と見なす行程を備える請求項37記載の方法。

【請求項39】 更に、非格子状に配置された白輪郭を再接続する行程を備える請求項37記載の方法。

【請求項40】 前記非テキスト要素は、前記再接続行程での再接続に基づいて表と見なされる請求項39記載の方法。

【請求項41】 更に、非テキスト要素内の白輪郭の充填率を算出する行程を備える請求項35記載の方法。

【請求項42】 非テキスト要素は、充填率が高い場合に表と見なされない請求項41記載の方法。

【請求項43】 画素画像データ内で文字のテキストファイルを形成する方法であって、画素画像データが2値の画素画像データでない場合に、画素画像データを2値の画素画像データに変換する行程を含む画素画像データの入力行程と、画素画像データ内で連結要素の輪郭を追跡する行程と、追跡された連結要素のサイズに基づいて連結要素がテキスト要素か非テキスト要素かを定める行程と、隣接するテキスト要素の近接関係に基づいてテキスト要素を選択的に横方向に連結してテキストラインを形成する行程と、隣接するテキストラインの近接関係とテキスト要素間の非テキスト要素の位置とに基づいてテキストラインを選択的に縦方向に連結してテキストブロックを形成する行程とを含み、画素画像データのブロックを選別する行程と、

4

前記輪郭追跡で追跡された画素要素に基づいて、階層の木構造を形成する行程と、

テキストブロックを画素画像データのラインに分割する行程と、

前記分割行程で分割されたラインから文字を切り出す行程と、

前記切り出行程で切り出された文字を認識し、前記認識に基づいて文字コードを割り当てる行程と、

前記階層の木構造で確定された順に対応して文字コードをコンピュータのテキストファイルに格納する行程とを備える方法。

【請求項44】 更に、画素画像データが前処理される前処理行程を備える請求項43記載の方法。

【請求項45】 前記前処理行程は画像圧縮行程を含む請求項44記載の方法。

【請求項46】 前記前処理行程は画素画像データの画質を向上する行程を含む請求項45記載の方法。

【請求項47】 更に、文脈チェックを含み、前記認識行程で認識された文字を後処理する行程を備える請求項43記載の方法。

【請求項48】 更に、非テキスト要素の内部で白輪郭を追跡する行程を備える請求項43記載の方法。

【請求項49】 更に、非テキストに識別子を付与する行程を更に備える請求項48記載の方法。

【請求項50】 更に、非テキスト要素に含まれる白輪郭の数に基づいて表識別子を付与する行程を備える請求項49記載の方法。

【請求項51】 更に、非格子状配置の白輪郭を再接続する行程を備え、非テキスト要素には前記再接続行程での再接続率に基づいて表の識別子が付与される請求項49記載の方法。

【請求項52】 更に、非テキスト要素内の白輪郭の充填率を算出する行程を備え、充填率が低い場合に非テキスト要素に表の識別子を付与する請求項49記載の方法。

【請求項53】 画素画像データから画素ブロックを選別する装置であって、画素データ内で連結要素の輪郭を追跡する輪郭追跡手段と、

前記輪郭追跡手段によって追跡された各連結要素を囲んで矩形を形成する矩形形成手段と、

サイズと他の矩形への近接関係とに基づいて矩形を選択的に横方向に連結して、ラインを形成する第1連結手段と、

サイズと他のラインへの近接関係に基づいてラインを選択的に縦方向に連結して、ブロックを形成する第2連結手段とを備える装置。

【請求項54】 更に、画素画像データを入力する入力手段を備え、

該入力手段は、画素画像データが2値画素画像データで

5

ない場合に、画素画像データを2値画素画像データに変換する変換手段を含む請求項53記載の装置。

【請求項55】 前記矩形形成手段では、各連結要素を囲む最小の矩形が形成される請求項53記載の装置。

【請求項56】 更に、前記矩形形成手段で形成された対応する矩形の位置に基づいて、前記輪郭追跡手段で追跡された連結要素の階層の木構造を形成する第2の形成手段を備える請求項53記載の装置。

【請求項57】 更に、前記矩形形成手段で形成された矩形をテキスト要素と非テキスト要素とに分類する分類手段を備える請求項56記載の装置。

【請求項58】 更に、テキスト要素のブロック内の文字画像を認識する認識手段を備える請求項57記載の装置。

【請求項59】 前記分類手段は、要素の高さの所定のしきい値に対応して分類する請求項57記載の装置。

【請求項60】 前記分類手段は、前記矩形形成手段で形成された矩形の高さに関連する統計情報に対応して分類する請求項57記載の装置。

【請求項61】 前記第1連結手段と第2連結手段とは、非テキスト要素には使用されない請求項57記載の装置。

【請求項62】 更に、非テキスト要素に対して白輪郭を得る手段を備える請求項57記載の装置。

【請求項63】 非テキスト要素は白輪郭の数に対応して表の要素と見なされる請求項62記載の装置。

【請求項64】 更に、白輪郭の充填率を算出する算出手段を備える請求項62記載の装置。

【請求項65】 非テキスト要素は、充填率が高い場合には画像データと見なされない請求項64記載の装置。

【請求項66】 更に、非格子状に配置された白輪郭を再接続する再接続手段を備える請求項64記載の装置。

【請求項67】 非テキスト要素は、再接続率が高くない場合に表と見なされる請求項66記載の装置。

【請求項68】 白輪郭は4方向で算出される請求項62記載の装置。

【請求項69】 連結要素の輪郭は、少なくとも8方向に追跡される請求項53記載の装置。

【請求項70】 前記輪郭追跡手段では、連結要素の外部のみで連結要素の輪郭が追跡される請求項53記載の装置。

【請求項71】 更に、画素画像データ内でギャップを検出する検出手段を備え、前記第1連結手段では、ギャップが矩形を隔てている場合には矩形がラインに連結されない請求項53記載の装置。

【請求項72】 矩形の間を縦に延びるギャップに対応して段が検出される請求項71記載の装置。

【請求項73】 前記第2連結手段は、前記第1連結手段で連結されたテキストデータのライン間に非テキストの境界を決定する決定手段を含み、前記第2連結手段

6

は、間に非テキストの境界がある場合にラインを縦方向にブロックとして連結しない請求項53記載の装置。

【請求項74】 更に、前記輪郭追跡手段の前に、画素画像データを圧縮する圧縮手段を備える請求項53記載の装置。

【請求項75】 画素画像データ内の文字を認識する装置であって、

画素画像データ内で連結要素の輪郭を追跡する行程と、定められた連結要素がテキスト要素を含むか非テキスト要素を含むかを定める行程と、テキスト要素を横方向に選択的に連結してテキストラインを形成する行程と、テキストラインを選択的に縦方向に連結してテキストブロックを形成する行程とを含み、画素画像データから画素ブロックを選別するブロック選別手段と、

テキストブロックを画素画像データのラインに分割する分割手段と、

前記分割手段で分割されたラインから文字を切り出す切出手段と、

前記切出手段で切り出された文字を認識する認識手段と、

前記認識手段で認識された文字を、前記ブロック選別手段で確定された順に対応して格納する格納手段とを備える装置。

【請求項76】 更に、テキスト要素のブロック内の文字画像を認識する第2の認識手段を備える請求項75記載の装置。

【請求項77】 更に、画素画像データが前処理される前処理手段を備える請求項75記載の装置。

【請求項78】 前記前処理手段は画像圧縮手段を含む請求項77記載の装置。

【請求項79】 前記前処理手段は画素画像データの画質を向上するフィルタ手段を含む請求項77記載の装置。

【請求項80】 更に、前記認識手段で認識された文字を後処理する後処理手段を備える請求項75記載の装置。

【請求項81】 前記後処理手段は文脈チェック手段を含む請求項80記載の装置。

【請求項82】 更に、画素画像データを入力する入力手段を備え、

該入力手段は、画素画像データが2値画素画像データでない場合に、画素画像データを2値画素画像データに変換する変換手段を含む請求項75記載の装置。

【請求項83】 前記ブロック選別手段は、前記輪郭追跡手段で追跡された連結要素に基づいて階層の木構造を形成する手段を含む請求項75記載の装置。

【請求項84】 更に、前記輪郭追跡手段で定められた連結要素を囲む矩形を形成する形成手段を備える請求項83記載の装置。

【請求項85】 前記階層の木構造は、前記矩形形成手

段で形成された矩形の位置に基づいて形成される請求項84記載の装置。

【請求項86】 更に、第1及び第2連結手段に基づいて、階層の木構造を更新する更新手段を備える請求項64記載の装置。

【請求項87】 更に、連結要素をテキスト要素又は非テキスト要素に分類する分類手段を備える請求項83記載の装置。

【請求項88】 前記第1及び第2連結手段は、テキスト要素を連結するが非テキスト要素は連結しない請求項87記載の装置。

【請求項89】 更に、非テキスト要素の内部で白輪郭を追跡する手段を備える請求項87記載の装置。

【請求項90】 更に、非テキスト要素に含まれる白輪郭の数に基づいて表と見なす手段を備える請求項89記載の装置。

【請求項91】 更に、非格子状に配置された白輪郭を再接続する再接続手段を備える請求項89記載の装置。

【請求項92】 前記非テキスト要素は、前記再接続手段での再接続に基づいて表と見なされる請求項91記載の装置。

【請求項93】 更に、非テキスト要素内の白輪郭の充填率を算出する算出手段を備える請求項87記載の装置。

【請求項94】 非テキスト要素は、充填率が高い場合に表と見なされない請求項93記載の装置。

【請求項95】 画素画像データ内で文字のテキストファイル形成する装置であって、

画素画像データが2値の画素画像データでない場合に、画素画像データを2値の画素画像データに変換する変換手段を含む画素画像データの入力手段と、

画素画像データ内で連結要素の輪郭を追跡する輪郭追跡手段と、取り出された連結要素のサイズに基づいて連結要素がテキスト要素か非テキスト要素かを決定する決定手段と、隣接するテキスト要素の近接関係に基づいてテキスト要素を選択的に横方向に連結してテキストラインを形成する第1連結手段と、隣接するテキストラインの近接関係とテキスト要素間の非テキスト要素の位置とに基づいてテキストラインを選択的に縦方向に連結してテキストブロックを形成する第2連結手段とを含み、画素画像データのブロックを選別するブロック選別手段と、

前記輪郭追跡で追跡された画素要素に基づいて、階層の木構造を形成する形成手段と、

テキストブロックを画素画像データのラインに分割する分割手段と、

前記分割手段で分割されたラインから文字を切り出す切出手段と、

前記切出手段で切り出された文字を認識し、前記認識に基づいて文字コードを割り当てる認識手段と、

前記階層の木構造で確定された順に対応して文字コード

をコンピュータのテキストファイルに格納する格納手段とを備える装置。

【請求項96】 更に、画素画像データが前処理される前処理手段を備える請求項95記載の装置。

【請求項97】 前記前処理手段は画像圧縮手段を含む請求項96記載の装置。

【請求項98】 前記前処理手段は画素画像データの画質向上手段を含む請求項97記載の装置。

【請求項99】 更に、文脈チェック手段を含み、前記認識手段で認識された文字を後処理する後処理手段を備える請求項95記載の装置。

【請求項100】 更に、非テキスト要素の内部で白輪郭を追跡する手段を備える請求項95記載の装置。

【請求項101】 更に、非テキストに識別子を付与する付与手段を更に備える請求項100記載の装置。

【請求項102】 更に、非テキスト要素に含まれる白輪郭の数に基づいて、非テキスト要素に表の識別子が付与される請求項101記載の装置。

【請求項103】 更に、非格子状配置の白輪郭を再接続する再接続手段を備え、非テキスト要素には前記再接続手段での再接続率に基づいて表の識別子が付与される請求項101記載の装置。

【請求項104】 更に、非テキスト要素内の白輪郭の充填率を算出する算出手段を備え、充填率が高くない場合に非テキスト要素に表識別子を付与する請求項101記載の装置。

【請求項105】 画素画像データでラインの位置を割り出す方法であって、

画像データの横方向に伸びた部分の全域で画像密度の水平投影を得る行程と、

前記水平投影に基づいて、画像密度が投影された部分の幅を減少するか否かを決定する行程と、

前記水平投影に基づいて、ラインの位置を割り出す行程を備える方法。

【請求項106】 前記決定行程は、実質的に空白の部分で水平投影を調べる行程と、実質的に空白の部分が見付からない場合に幅を減少する行程とを備える請求項105記載の方法。

【請求項107】 前記調べる行程は、間隔の小さい非空白の部分を選択する行程を含む請求項106記載の方法。

【請求項108】 前記水平投影を得る行程と決定する行程とは、順に繰り返して実行される請求項105記載の方法。

【請求項109】 前記割り出行程では、水平投影の実質的に空白の領域間の画像データのライン位置を、ラインとして割り出す請求項105記載の方法。

【請求項110】 更に、画素画像データを前処理する前処理行程を備える請求項105記載の方法。

【請求項111】 段に分割可能な画素画像データ内で

ラインの位置を割り出す方法であって、画像データの少なくとも1つの横方向に伸びる段の全域で、画像密度の水平投影を得る行程と、前記行程で得られた水平投影に基づいて、画像密度が投影される段の数を増加するか否かを決定する行程と、前記水平投影に基づいてデジタル画像データ内でラインの位置を割り出す行程とを備える方法。

【請求項112】 前記決定行程は、実質的に空白の部分で水平投影を調べる行程と、実質的に空白の部分が見付からない場合に、段の数を増加する行程とを備える請求項111記載の方法。

【請求項113】 前記調べる行程は間隔の小さい非空白の部分で連結する行程を含む請求項112記載の方法。

【請求項114】 前記水平投影を得る行程と決定する行程とは、順に繰り返して実行される請求項111記載の方法。

【請求項115】 前記割出行程では、水平投影の実質的に空白の領域間の画像データのライン位置を、ラインとして割り出す請求項111記載の方法。

【請求項116】 更に、画像データの隣接する段のラインが重なるか否かを決定する行程を備える請求項111記載の方法。

【請求項117】 前記ラインが重なるかを決定する行程は、スケール不変の決定から成る請求項116記載の方法。

【請求項118】 更に、画素画像データを前処理する前処理行程を備え、前記前処理行程は画像縮小行程を含む請求項111記載の方法。

【請求項119】 前記画像縮小は水平方向と垂直方向とで異なる請求項118記載の方法。

【請求項120】 画素画像データのラインセグメントから文字画像を切り出す方法であって、ラインセグメントから非接触で重なっていない文字を切り出す第1の切出行程と、ラインセグメントから接触文字を切り出す第2の切出行程とを備える方法。

【請求項121】 更に、高解像度の画像セグメントを低解像度に圧縮する画像データのイメージセグメントの前処理行程を備える請求項120記載の方法。

【請求項122】 垂直方向と水平方向とで異なる圧縮率が使われる請求項121記載の方法。

【請求項123】 更に、前記第1の切出行程で切り出された文字を認識処理する行程を備え、前記第2の切出行程は前記認識行程で認識されなかった文字に対してのみ実行される請求項120記載の方法。

【請求項124】 更に、非接触で重なっている文字データをラインセグメントから切り出す中間の切出行程を備える請求項123記載の方法。

【請求項125】 前記中間切出行程は、非接触で重な

っている文字を画素データの輪郭の追跡により切り出す行程を含む請求項124記載の方法。

【請求項126】 前記中間切出行程は、間違っ切り出された文字を再接続する行程を含む請求項124記載の方法。

【請求項127】 更に、前記認識行程で認識されなかった文字を再接続する行程を備える請求項123記載の方法。

【請求項128】 前記第1の切出行程は、前記ラインセグメントを非空白の画素データが見付かるまでとびとびに処理する行程を含む請求項120記載の方法。

【請求項129】 更に、前記とびとびの処理行程で非空白の画素が見付かった場合に、厳密に前後をサーチする行程を備える請求項128記載の方法。

【請求項130】 前記第2の切出行程は非垂直切り出しによって接触文字を切り出す行程を含む請求項120記載の方法。

【請求項131】 非垂直切り出しの角度は画素密度の垂直投影特性に対応して決められる請求項130記載の方法。

【請求項132】 更に、垂直投影特性に基づいて少なくとも1つの回転された投影特性を得る行程を備えることを特徴とする請求項131記載の方法。

【請求項133】 更に、前記垂直投影特性の角度に近接する角度で複数の回転された投影特性を得る行程を備える請求項132記載の方法。

【請求項134】 非垂直切り出しの角度は、前記回転投影特性を得る行程と垂直投影特性を得る行程とから得られた投影特性の最小値に基づいて得られる請求項133記載の方法。

【請求項135】 前記第2の切出行程は、前記ラインセグメント内の文字の特徴が既知かどうかに基づいて選択可能である請求項120記載の方法。

【請求項136】 前記第2の切出行程は、前記文字セット内の文字データのスペースの統計に基づいて切り出す行程を含む請求項135記載の方法。

【請求項137】 更に、小さな文字のセットを再接続する行程を備える請求項136記載の方法。

【請求項138】 文字画素データのラインセグメント内で接触文字を切り出す方法であって、

ラインセグメント内で画素データの垂直投影特性を得る行程と、

垂直投影特性の最小値から隣接する最大値へ方向から少なくとも1つの角度を算出する第1の算出行程と、

前記第1の算出行程で算出された角度に基づいて、少なくとも1つの回転された投影特性を算出する第2の算出行程と、

前記回転投影特性の最小値に対応する位置で前記回転投影特性の角度にラインセグメントを切る行程とを備える

方法。

【請求項139】 前記第1の算出行程は、前記垂直投影特性を複数のしきい値と比較して前記最小値と隣接する最大値との位置を決定する行程を含む請求項138記載の方法。

【請求項140】 更に、最小値と隣接する最大値とが見付からない場合に、前記しきい値を増加させる行程を備える請求項139記載の方法。

【請求項141】 最小値は、少なくとも一方の側が第2しきい値より大きい最大値によって囲まれ、第1しきい値より小さな垂直投影特性上の位置として割り出される請求項139記載の方法。

【請求項142】 更に、各々が前記第2の算出行程で算出された回転投影特性に近接する角度で算出される複数の回転投影特性を算出する行程を備える請求項138記載の方法。

【請求項143】 更に、複数の回転投影特性と垂直投影特性との全体の最小値を割り出す行程を備え、前記切断行程は最小値の位置に対応する位置で最小値が得られた角度に対応する角度に切る請求項142記載の方法。

【請求項144】 画素画像データ内で文字を認識する  
20 方法であって、

画素画像データから画素ブロックを選別する行程と、  
少なくとも1つの段全体の画素密度の水平投影に基づいてテキストブロックを少なくとも1つの段に適応的に分配することにより、テキストブロックを画素データのラインに分割する行程と、

非接触で重なっていない文字がラインセグメントから切り出される第1の切出行程と、接触文字を切り出す少なくとも1つの追加の切出行程とを含み、前記分割行程で分割されたラインセグメントから文字を切り出す行程  
30 と、

前記文字切出行程で切り出された文字を認識する行程と、

前記認識行程で認識された文字を前記ブロック選別行程で確定された順に対応して格納する行程とを備える方法。

【請求項145】 更に、画素画像データが2値の画素画像データでない場合に、画素画像データを2値の画素画像データに変換する行程を含む画素画像データを入力する行程を備える請求項144記載の方法。

【請求項146】 更に、画素画像データが前処理される前処理行程を備える請求項144記載の方法。

【請求項147】 前記前処理行程は、画像圧縮行程を含む請求項146記載の方法。

【請求項148】 前記前処理行程は、画素画像データの画質を向上するフィルタ行程を含む請求項146記載の方法。

【請求項149】 更に、前記認識行程で認識された文字を後処理する行程を備える請求項144記載の方法。

【請求項150】 前記後処理は文脈チェックを含む請求  
50

項149記載の方法。

【請求項151】 前記第1の切出行程はラインセグメントをとびとびに処理する行程を含む請求項144記載の方法。

【請求項152】 前記追加の切出行程は、画素画像データの文字について特徴が既知か否かに対応して選択可能である請求項144記載の方法。

【請求項153】 前記追加の切出行程では既知の文字セットの統計に基づいて切り出しを行う請求項152記載の方法。

【請求項154】 前記追加の切出行程では画素密度の回転投影に対応して切り出しを行う請求項152記載の方法。

【請求項155】 画素密度の回転投影は画素密度の垂直投影に基づいて決められた角度で得られる請求項154記載の方法。

【請求項156】 更に、前記第1の切出行程と前記少なくとも1つの追加の切出行程の間に中間の切出行程を備える請求項154記載の方法。

【請求項157】 前記中間の切出行程はラインセグメント内の重なった画素画像の輪郭を追跡する行程を含む請求項156記載の方法。

【請求項158】 更に、文字が間違っって切り出された場合に、前記中間の切出行程で切り出された文字を再接続する行程を備える請求項157記載の方法。

【請求項159】 前記認識行程は、前記追加の切出行程の前に前記第1の切出行程で切り出された文字を認識し、前記追加の切出行程は前記認識行程で認識されない文字に対して行なわれる請求項158記載の方法。

【請求項160】 更に、文字が前記認識行程で認識されない場合に、前記少なくとも1つの追加の切出行程で切り出された文字を再接続する行程を備える請求項159記載の方法。

【請求項161】 画素画像データから文字のテキストファイルを形成する方法であって、

画素画像データが2値の画素画像データでない場合に、画素画像データを2値の画素画像データに変換する行程を含む画素画像データの入力行程と、

画素画像データのブロックがテキストブロックか非テキストブロックかに対応して画素画像データのブロックを選別する行程と、

テキスト及び非テキストブロックの階層的木構造を得る行程と、

少なくとも1つの段の全域の画素密度の水平投影に基づいて、テキストブロックを前記少なくとも1つの段で適応的に分離することにより、テキストブロックを画素画像データのラインに分割する行程と、

非接触で重なっていない文字をラインセグメントから切り出す第1の切出行程と、接触文字を切り出す少なくとも1つの追加の切出行程とを含み、前記分割行程で分割



されたラインから文字を切り出す行程とを備える方法。

【請求項162】 更に、画素画像データが前処理される前処理行程を備える請求項161記載の方法。

【請求項163】 前記前処理行程は画像圧縮行程を含む請求項162記載の方法。

【請求項164】 前記前処理行程は画素画像データの画質を向上する行程を含む請求項163記載の方法。

【請求項165】 更に、文脈チェックを含み、前記認識行程で認識された文字を後処理する行程を備える請求項161記載の方法。

【請求項166】 前記第1の切出行程はラインセグメントをとびとびに処理する行程を含む請求項161記載の方法。

【請求項167】 前記追加の切出行程は、画素画像データの文字について特徴が既知か否かに対応して選択可能である請求項166記載の方法。

【請求項168】 前記追加の切出行程では既知の文字セットの統計に基づいて切り出しを行う請求項167記載の方法。

【請求項169】 前記追加の切出行程では画素密度の回転投影に対応して切り出しを行う請求項167記載の方法。

【請求項170】 画素密度の回転投影は画素密度の垂直投影に基づいて決められた角度で得られる請求項169記載の方法。

【請求項171】 更に、前記第1の切出行程と前記少なくとも1つの追加の切出行程の間に中間の切出行程を備える請求項161記載の方法。

【請求項172】 前記中間の切出行程はラインセグメント内の重なった画素画像の輪郭を追跡する行程を含む請求項171記載の方法。

【請求項173】 更に、文字が間違っって切り出された場合に、前記中間の切出行程で切り出された文字を再接続する行程を備える請求項172記載の方法。

【請求項174】 前記認識行程は、前記追加の切出行程の前に前記第1の切出行程で切り出された文字を認識し、前記追加の切出行程は前記認識行程で認識されない文字に対して行なわれる請求項173記載の方法。

【請求項175】 更に、文字が前記認識行程で認識されない場合に、前記少なくとも1つの追加の切出行程で切り出された文字を再接続する行程を備える請求項174記載の方法。

【請求項176】 画素画像データから文字のテキストファイルを形成する方法であって、画素画像データが2値の画素画像データでない場合に、画素画像データを2値の画素画像データに変換する行程を含み、画素画像データを入力する行程と、

画素画像データ内で連結要素の輪郭を追跡する行程と、前記追跡行程の間に追跡された連結要素のサイズに基づいて追跡された連結要素がテキスト要素を含むか非テキ

スト要素を含むから決める行程と、隣接するテキスト要素の近接関係に基づいてテキスト要素を選択的に横方向に連結してテキストラインを形成する行程と、隣接するテキストラインの近接関係とテキストライン間の非テキスト要素の位置とに基づいてテキストラインを選択的に縦方向に連結する行程とを含み、画素画像データのブロックを選別する行程と、

前記輪郭追跡行程で追跡された連結要素に基づいて、階層の木構造を形成する行程と、

10 少なくとも1つの段の全域での画素密度の水平投影に基づいて、テキストブロックを前記少なくとも1つの段に適応的に分離することによって、テキストブロックを画素データのラインに分割する行程と、

非接触で重なっていない文字をラインセグメントから切り出す第1の切出行程と、接触文字を切り出す少なくとも1つの追加の切出行程とを含み、前記分割行程で分割されたラインから文字を切り出す行程と、

前記文字切出行程で切り出された文字を認識し、前記認識に基づいて文字コードを割り付ける行程と、

20 階層の木構造により確定された順に対応して、文字コードをコンピュータのテキストファイルに格納する行程とを備える方法。

【請求項177】 更に、画素画像データが前処理される前処理行程を備える請求項176記載の方法。

【請求項178】 前記前処理行程は画像圧縮行程を含む請求項177記載の方法。

【請求項179】 前記前処理行程は画素画像データの画質を向上する行程を含む請求項178記載の方法。

【請求項180】 更に、文脈チェックを含み、前記認識行程で認識された文字を後処理する行程を備える請求項176記載の方法。

【請求項181】 前記第1の切出行程はラインセグメントをとびとびに処理する行程を含む請求項176記載の方法。

【請求項182】 前記追加の切出行程は、画素画像データの文字について特徴が既知か否かに対応して選択可能である請求項176記載の方法。

【請求項183】 前記追加の切出行程では既知の文字セットの統計に基づいて切り出しを行う請求項182記載の方法。

40 【請求項184】 前記追加の切出行程では画素密度の回転投影に対応して切り出しを行う請求項182記載の方法。

【請求項185】 画素密度の回転投影は画素密度の垂直投影に基づいて決められた角度で得られる請求項184記載の方法。

【請求項186】 更に、前記第1の切出行程と前記少なくとも1つの追加の切出行程の間に中間の切出行程を備える請求項176記載の方法。

50 【請求項187】 前記中間の切出行程はラインセグメ

ント内の重なった画素画像の輪郭を追跡する行程を含む請求項186記載の方法。

【請求項188】 更に、文字が間違っ切り出された場合に、前記中間の切出行程で切り出された文字を再接続する行程を備える請求項187記載の方法。

【請求項189】 前記認識行程は、前記追加の切出行程の前に前記第1の切出行程で切り出された文字を認識し、前記追加の切出行程は前記認識行程で認識されない文字に対して行なわれる請求項188記載の方法。

【請求項190】 更に、文字が前記認識行程で認識されない場合に、前記少なくとも1つの追加の切出行程で切り出された文字を再接続する行程を備える請求項189記載の方法。

【請求項191】 更に、非テキスト要素の内部で白輪郭を追跡する行程を備える請求項176記載の方法。

【請求項192】 更に、非テキスト要素に識別子を付加する行程を備える請求項191記載の方法。

【請求項193】 非テキスト要素内に含まれる白輪郭の数に基づいて非テキスト要素を表と見なす請求項192記載の方法。

【請求項194】 更に、非格子状配置の白輪郭を再接続する行程を備え、非テキスト要素には前記再接続行程での再接続率に基づいて表識別子が付加される請求項192記載の方法。

【請求項195】 更に、非テキスト要素内の白輪郭の充填率を算出する行程を備え、充填率が低い場合に非テキスト要素を表識別子を付加する請求項192記載の方法。

【請求項196】 画素画像データでラインの位置を割り出す装置であって、画像データの横方向に伸びた部分の全域で画像密度の水平投影を得る手段と、水平投影に基づいて、画像密度が投影された部分の幅を減少するか否かを決定する決定行程と、水平投影に基づいて、ラインの位置を割り出す割出行程とを備える装置。

【請求項197】 前記決定手段は、実質的に空白の部分で水平投影を調べる手段と、実質的に空白の部分が見付からない場合に幅を減少する減少手段とを備える請求項196記載の装置。

【請求項198】 前記調べる手段は間隔の小さい非空白の部分に接続する接続手段を含む請求項197記載の装置。

【請求項199】 前記水平投影を得る手段と決定手段とは、順に繰り返して使用される請求項196記載の装置。

【請求項200】 前記割出手段では、水平投影の実質的に空白の領域間の画像データのライン位置を、ラインとして割り出す請求項196記載の装置。

【請求項201】 更に、画素画像データを前処理する

前処理手段を備える請求項196記載の装置。

【請求項202】 段に分離可能な画素画像データ内でラインの位置を割り出す装置であって、画像データの少なくとも1つの横方向に伸びる段の全域で、画像密度の水平投影を得る手段と、前記手段で得られた水平投影に基づいて、画像密度が投影される段の数を増加するか否かを決定する決定手段と、

前記水平投影に基づいてデジタル画像データ内でラインの位置を割り出す割出手段とを備える装置。

【請求項203】 前記決定手段は、実質的に空白の部分で水平投影を調べる手段と、実質的に空白の部分が見付からない場合に段の数を増加する増加手段とを備える請求項202記載の装置。

【請求項204】 前記調べる手段は間隔の小さい非空白の部分に連結する連結手段を含む請求項203記載の装置。

【請求項205】 前記水平投影を得る手段と決定手段とは、順に繰り返して使用される請求項202記載の装置。

【請求項206】 前記割出手段では、水平投影の実質的に空白の領域間の画像データのライン位置を、ラインとして割り出す請求項202記載の装置。

【請求項207】 更に、画像データの隣接する段のラインが重なるか否かを定める第2の決定手段を備える請求項202記載の装置。

【請求項208】 前記第2の決定手段は、画像データの隣接する段のブロックが重なるか否かを決定するスケール不変の決定手段から成る請求項207記載の装置。

【請求項209】 更に、画素画像データを前処理する前処理手段を備え、前記前処理手段は画像縮小手段を含む請求項202記載の装置。

【請求項210】 前記画像縮小は水平方向と垂直方向とで異なる請求項209記載の装置。

【請求項211】 画素画像データのラインセグメントから文字画像を切り出す装置であって、ラインセグメントから非接触で重なっていない文字を切り出す第1の切出手段と、ラインセグメントから接触文字を切り出す第2の切出手段とを備える装置。

【請求項212】 更に、高解像度の画像セグメントを低解像度に圧縮するように画像データのイメージセグメントを前処理する前処理手段を備える請求項211記載の装置。

【請求項213】 垂直方向と水平方向とで異なる圧縮率が使われる請求項212記載の装置。

【請求項214】 更に、前記第1の切出手段で切り出された文字を認識処理する認識手段を備え、前記第2の切出手段は前記認識手段で認識されなかった文字に対してのみ使用される請求項212記載の装置。



【請求項215】 更に、非接触で重なった文字データをラインセグメントから中間で切り出す第3の切出手段を備える請求項214記載の装置。

【請求項216】 前記第3の切出手段は、非接触で重なった文字を画素データの輪郭を追跡する輪郭追跡手段を含む請求項215記載の装置。

【請求項217】 前記第3の切出手段は、間違って切り出された文字を再接続する再接続手段を含む請求項215記載の装置。

【請求項218】 更に、前記認識手段で認識されなかった文字を再接続する第2の再接続手段を備える請求項214記載の装置。

【請求項219】 前記第1の切出手段は、前記ラインセグメントを非空白の画素データが見付かるまでとびとびに処理する手段を含む請求項211記載の装置。

【請求項220】 更に、前記とびとびの処理手段で非空白の画素が見付かった場合に、緻密に前後をサーチする検索手段を備える請求項219記載の装置。

【請求項221】 前記第2の切出手段は非垂直切り出しによって接触文字を切り出す請求項213記載の装置。

【請求項222】 非垂直切り出しの角度は画素密度の垂直投影特性に対応して決められる請求項221記載の装置。

【請求項223】 更に、垂直投影特性に基づいて少なくとも1つの回転された投影特性を得る手段を備えることを特徴とする請求項222記載の装置。

【請求項224】 更に、前記垂直投影特性の角度に近接する角度で複数の回転された投影特性を得る手段を備える請求項223記載の装置。

【請求項225】 非垂直切り出しの角度は、前記前記回転投影特性を得る手段と垂直投影特性を得る手段とから得られた投影特性の最小値に基づいて得られる請求項224記載の装置。

【請求項226】 前記第2の切出手段は、前記ラインセグメント内の文字の特徴が既知かどうかに基づいて選択可能である請求項211記載の装置。

【請求項227】 前記第2の切出手段は、前記文字セット内の文字データのスペースの統計に基づいて切り出す請求項226記載の装置。

【請求項228】 更に、小さな文字のセットを再接続する第3の再接続手段を備える請求項227記載の装置。

【請求項229】 文字画素データのラインセグメント内で接触文字を切り出す装置であって、ラインセグメント内で画素データの垂直投影特性を得る手段と、

垂直投影特性の最小値から隣接する最大値へ方向から少なくとも1つの角度を算出する第1の算出手段と、

前記第1の算出手段で算出された角度に基づいて、少な

くとも1つの回転された投影特性を算出する第2の算出手段と、

前記回転投影特性の最小値に対応する位置で前記回転投影特性の角度にラインセグメントを切る切断手段とを備える装置。

【請求項230】 前記第1の算出手段は、前記垂直投影特性を複数のしきい値と比較して前記最小値と隣接する最大値との位置を決定する比較手段を含む請求項229記載の装置。

【請求項231】 更に、最小値と隣接する最大値とが見付からない場合に、前記しきい値を増加させる増加手段を備える請求項230記載の装置。

【請求項232】 最小値は、少なくとも一方の側が第2しきい値より大きい最大値によって囲まれ、第1しきい値より小さな垂直投影特性上の位置として割り出される請求項231記載の装置。

【請求項233】 更に、各々が前記第2の算出手段で算出された回転投影特性に近接する角度で算出される複数の回転投影特性を算出する第3の算出手段を備える請求項229記載の装置。

【請求項234】 更に、複数の回転投影特性と垂直投影特性との全体の最小値を割り出す割出手段を備え、前記切断手段は最小値の位置に対応する位置で最小値が得られた角度に対応する角度に切る請求項233記載の装置。

【請求項235】 画素画像データ内で文字を認識する装置であって、画素画像データから画素ブロックを選別する選別手段と、

少なくとも1つの段全体の画素密度の水平投影に基づいてテキストブロックを少なくとも1つの段に適応的に分配することにより、テキストブロックを画素データのラインに分割する分割手段と、

非接触で重なっていない文字がラインセグメントから切り出される第1の切出手段と、接触文字を切り出す少なくとも1つの追加の切出手段とを含み、前記分割手段で分割されたラインセグメントから文字を切り出す切出手段と、

前記切出手段で切り出された文字を認識する認識手段と、

前記認識手段で認識された文字を前記選別手段で確定された順に対応して格納する格納手段とを備える装置。

【請求項236】 更に、画素画像データが2値の画素画像データでない場合に、画素画像データを2値の画素画像データに変換する変換手段を含む画素画像データの入力手段を備える請求項235記載の装置。

【請求項237】 更に、画素画像データが前処理される前処理手段を備える請求項235記載の装置。

【請求項238】 前記前処理手段は画像圧縮手段を含む請求項237記載の装置。

【請求項239】 前記前処理手段は画素画像データの画質を向上するフィルタ手段を含む請求項237記載の装置。

【請求項240】 更に、前記認識手段で認識された文字を後処理する後処理手段を備える請求項235記載の装置。

【請求項241】 前記後処理手段は文脈チェック手段を含む請求項240記載の装置。

【請求項242】 前記第1の切出手段はラインセグメントをとびとびに処理する手段を含む請求項235記載の装置。

【請求項243】 前記追加の切出手段は、画素画像データの文字について特徴が既知か否かに対応して選択可能である請求項235記載の装置。

【請求項244】 前記追加の切出手段は既知の文字セットの統計に基づいて切り出しを行う請求項243記載の装置。

【請求項245】 前記追加の切出手段は画素密度の回転投影に対応して切り出しを行う請求項243記載の装置。

【請求項246】 画素密度の回転投影は画素密度の垂直投影に基づいて決められた角度で得られる請求項245記載の装置。

【請求項247】 更に、前記第1の切出手段の使用後で前記少なくとも1つの追加の切出手段の使用前に、中間の階層を切り出す中間の切出手段を備える請求項245記載の装置。

【請求項248】 前記中間の切出手段はラインセグメント内の重なった画素画像の輪郭を追跡する輪郭追跡手段を含む請求項247記載の装置。

【請求項249】 更に、文字が間違っって切り出された場合に、前記中間の切出手段で切り出された文字を再接続する再接続手段を備える請求項248記載の装置。

【請求項250】 前記認識手段は、前記追加の切出手段の前に前記第1の切出手段で切り出された文字を認識し、前記追加の切出手段は前記認識手段で認識されない文字に対して行なわれる請求項249記載の装置。

【請求項251】 更に、文字が前記認識手段で認識されない場合に、前記少なくとも1つの追加の切出手段で切り出された文字を再接続する第2の再接続手段を備える請求項250記載の装置。

【請求項252】 画素画像データから文字のテキストファイルを形成する装置であって、画素画像データが2値の画素画像データでない場合に、画素画像データを2値の画素画像データに変換する変換手段を含む画素画像データの入力手段と、画素画像データのブロックがテキストブロックか非テキストブロックかに対応して画素画像データのブロックを選別する選別手段と、テキスト及び非テキストブロックの階層的木構造を得る

手段と、

少なくとも1つの段の全域の画素密度の水平投影に基づいて、テキストブロックを前記少なくとも1つの段に適応的に分配することにより、テキストブロックを画素画像データのラインに分割する分割手段と、

非接触で重なっていない文字をラインセグメントから切り出す第1の切出手段と、接触文字を切り出す少なくとも1つの追加の切出手段とを含み、前記分割手段で分割されたラインから文字を切り出す切出手段と、

前記切出手段で切り出された文字を認識し、前記認識に基づいて文字コードを割り付ける認識手段と、前記階層的木構造により確立された順に対応して文字コードが格納されるコンピュータのテキストファイルとを備える装置。

【請求項253】 更に、画素画像データが前処理される前処理手段を備える請求項252記載の装置。

【請求項254】 前記前処理手段は画像圧縮手段を含む請求項253記載の装置。

【請求項255】 前記前処理手段は画素画像データの画質を向上する画質向上手段を含む請求項253記載の装置。

【請求項256】 更に、文脈チェック手段を含み、前記認識手段で認識された文字を後処理する後処理手段を備える請求項252記載の装置。

【請求項257】 前記第1の切出手段はラインセグメントをとびとびに処理する手段を含む請求項252記載の装置。

【請求項258】 前記追加の切出手段は、画素画像データの文字について特徴が既知か否かに対応して選択可能である請求項252記載の装置。

【請求項259】 前記追加の切出手段は既知の文字セットの統計に基づいて切り出しを行う請求項258記載の装置。

【請求項260】 前記追加の切出手段は画素密度の回転投影に対応して切り出しを行う請求項258記載の装置。

【請求項261】 画素密度の回転投影は画素密度の垂直投影に基づいて決められた角度で得られる請求項260記載の装置。

【請求項262】 更に、前記第1の切出手段と前記少なくとも1つの追加の切出手段の間に中間の階層を切り出す中間の切出手段を備える請求項252記載の装置。

【請求項263】 前記中間の切出手段はラインセグメント内の重なった画素画像の輪郭を追跡する輪郭追跡手段を含む請求項262記載の装置。

【請求項264】 更に、文字が間違っって切り出された場合に、前記中間の切出手段で切り出された文字を再接続する再接続手段を備える請求項263記載の装置。

【請求項265】 前記認識手段は、前記追加の切出手段の前に前記第1の切出手段で切り出された文字を認識

し、前記追加の切出手段は前記認識手段で認識されない文字に対して行なわれる請求項264記載の装置。

【請求項266】 更に、文字が前記認識手段で認識されない場合に、前記少なくとも1つの追加の切出手段で切り出された文字を再接続する第2の再接続手段を備える請求項265記載の装置。

【請求項267】 画素画像データから文字のテキストファイルを形成する装置であって、

画素画像データが2値の画素画像データでない場合に、画素画像データを2値の画素画像データに変換する変換手段を含み、画素画像データを入力する入力手段と、

画素画像データ内で連結要素の輪郭を追跡する輪郭追跡手段と、追跡された連結要素のサイズに基づいて追跡された連結要素がテキスト要素を含むか非テキスト要素を含むから決める決定手段と、隣接するテキスト要素の近接関係に基づいてテキスト要素を選択的に横方向に連結してテキストラインを形成する第1の連結手段と、隣接するテキストラインの近接関係とテキストライン間の非テキスト要素の位置とに基づいてテキストラインを選択的に縦方向に連結する第2の連結手段とを含み、画素画像データのブロックを選別する選別手段と、

前記輪郭追跡手段で追跡された連結要素に基づいて、階層の木構造を形成する形成手段と、

少なくとも1つの段の全域での画素密度の水平投影に基づいて、テキストブロックを前記少なくとも1つの段に適応的に分配することによって、テキストブロックを画素データのラインに分割する分割手段と、

非接触で重ならない文字をラインセグメントから切り出す第1の切出手段と、接触文字を切り出す少なくとも1つの追加の切出手段とを含み、前記分割手段で分割されたラインから文字を切り出す切出手段と、

前記文字切出手段で切り出された文字を認識し、前記認識に基づいて文字コードを割り付ける認識手段と、階層の木構造により確定された順に対応して文字コードをコンピュータのテキストファイルに格納する格納手段とを備える装置。

【請求項268】 更に、画素画像データが前処理される前処理手段を備える請求項267記載の装置。

【請求項269】 前記前処理手段は画像圧縮手段を含む請求項268記載の装置。

【請求項270】 前記前処理手段は画素画像データの画質を向上する画質向上手段を含む請求項269記載の装置。

【請求項271】 更に、文脈チェック手段を含み、前記認識手段で認識された文字を後処理する後処理手段を備える請求項267記載の装置。

【請求項272】 前記第1の切出手段はラインセグメントをとびとびに処理する手段を含む請求項267記載の装置。

【請求項273】 前記追加の切出手段は画素画像デー

タの文字について特徴が既知か否かに対応して選択可能である請求項267記載の装置。

【請求項274】 前記追加の切出手段は既知の文字セットの統計に基づいて切り出しを行う請求項273記載の装置。

【請求項275】 前記追加の切出手段は画素密度の回転投影に対応して切り出しを行う請求項273記載の装置。

【請求項276】 画素密度の回転投影は画素密度の垂直投影に基づいて決められた角度で得られる請求項275記載の装置。

【請求項277】 更に、前記第1の切出手段の後で前記少なくとも1つの追加の切出手段の前に中間の階層を切り出す中間の切出手段を備える請求項267記載の装置。

【請求項278】 前記中間の切出手段はラインセグメント内の重なった画素画像の輪郭を追跡する輪郭追跡手段を含む請求項277記載の装置。

【請求項279】 更に、文字が間違って切り出された場合に、前記中間の切出手段で切り出された文字を再接続する第1の再接続手段を備える請求項278記載の装置。

【請求項280】 前記認識手段は、前記追加の切出手段の前に前記第1の切出手段で切り出された文字を認識し、前記追加の切出手段は前記認識手段で認識されない文字に対して行なわれる請求項279記載の装置。

【請求項281】 更に、文字が前記認識手段で認識されない場合に、前記少なくとも1つの追加の切出手段で切り出された文字を再接続する第2の再接続手段を備える請求項280記載の装置。

【請求項282】 更に、非テキスト要素の内部で白輪郭を追跡する手段を備える請求項267記載の装置。

【請求項283】 更に、非テキスト要素に識別子を付加する付加手段を備える請求項282記載の装置。

【請求項284】 非テキスト要素内に含まれる白輪郭の数に基づいて非テキスト要素を表と見なす請求項283記載の装置。

【請求項285】 更に、非格子状配置の白輪郭を再接続する手段を備え、非テキスト要素には前記再接続手段での再接続率に基づいて表の識別子が付加される請求項283記載の装置。

【請求項286】 更に、非テキスト要素内の白輪郭の充填率を算出する算出手段を備え、充填率が低い場合に非テキスト要素に表の識別子を付加する請求項283記載の装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、文字認識方法及び装置に関わるものであり、特に認識処理に先立って、画像データに基づいて画像データブロックを分類・選別する方

し、前記追加の切出手段は前記認識手段で認識されない文字に対して行なわれる請求項264記載の装置。

【請求項266】 更に、文字が前記認識手段で認識されない場合に、前記少なくとも1つの追加の切出手段で切り出された文字を再接続する第2の再接続手段を備える請求項265記載の装置。

【請求項267】 画素画像データから文字のテキストファイルを形成する装置であって、

画素画像データが2値の画素画像データでない場合に、画素画像データを2値の画素画像データに変換する変換手段を含み、画素画像データを入力する入力手段と、画素画像データ内で連結要素の輪郭を追跡する輪郭追跡手段と、追跡された連結要素のサイズに基づいて追跡された連結要素がテキスト要素を含むか非テキスト要素を含むから決める決定手段と、隣接するテキスト要素の近接関係に基づいてテキスト要素を選択的に横方向に連結してテキストラインを形成する第1の連結手段と、隣接するテキストラインの近接関係とテキストライン間の非テキスト要素の位置とに基づいてテキストラインを選択的に縦方向に連結する第2の連結手段とを含み、画素画像データのブロックを選別する選別手段と、前記輪郭追跡手段で追跡された連結要素に基づいて、階層的木構造を形成する形成手段と、少なくとも1つの段の全域での画素密度の水平投影に基づいて、テキストブロックを前記少なくとも1つの段に適応的に分配することによって、テキストブロックを画素データのラインに分割する分割手段と、非接触で重ならない文字をラインセグメントから切り出す第1の切出手段と、接触文字を切り出す少なくとも1つの追加の切出手段とを含み、前記分割手段で分割されたラインから文字を切り出す切出手段と、前記文字切出手段で切り出された文字を認識し、前記認識に基づいて文字コードを割り付ける認識手段と、階層的木構造により確定された順に対応して文字コードをコンピュータのテキストファイルに格納する格納手段とを備える装置。

【請求項268】 更に、画素画像データが前処理される前処理手段を備える請求項267記載の装置。

【請求項269】 前記前処理手段は画像圧縮手段を含む請求項268記載の装置。

【請求項270】 前記前処理手段は画素画像データの画質を向上する画質向上手段を含む請求項269記載の装置。

【請求項271】 更に、文脈チェック手段を含み、前記認識手段で認識された文字を後処理する後処理手段を備える請求項267記載の装置。

【請求項272】 前記第1の切出手段はラインセグメントをとびとびに処理する手段を含む請求項267記載の装置。

【請求項273】 前記追加の切出手段は画素画像デー

タの文字について特徴が既知か否かに対応して選択可能である請求項267記載の装置。

【請求項274】 前記追加の切出手段は既知の文字セットの統計に基づいて切り出しを行う請求項273記載の装置。

【請求項275】 前記追加の切出手段は画素密度の回転投影に対応して切り出しを行う請求項273記載の装置。

【請求項276】 画素密度の回転投影は画素密度の垂直投影に基づいて決められた角度で得られる請求項275記載の装置。

【請求項277】 更に、前記第1の切出手段の後で前記少なくとも1つの追加の切出手段の前に中間の階層を切り出す中間の切出手段を備える請求項267記載の装置。

【請求項278】 前記中間の切出手段はラインセグメント内の重なった画素画像の輪郭を追跡する輪郭追跡手段を含む請求項277記載の装置。

【請求項279】 更に、文字が間違って切り出された場合に、前記中間の切出手段で切り出された文字を再接続する第1の再接続手段を備える請求項278記載の装置。

【請求項280】 前記認識手段は、前記追加の切出手段の前に前記第1の切出手段で切り出された文字を認識し、前記追加の切出手段は前記認識手段で認識されない文字に対して行なわれる請求項279記載の装置。

【請求項281】 更に、文字が前記認識手段で認識されない場合に、前記少なくとも1つの追加の切出手段で切り出された文字を再接続する第2の再接続手段を備える請求項280記載の装置。

【請求項282】 更に、非テキスト要素の内部で白輪郭を追跡する手段を備える請求項267記載の装置。

【請求項283】 更に、非テキスト要素に識別子を付加する付加手段を備える請求項282記載の装置。

【請求項284】 非テキスト要素内に含まれる白輪郭の数に基づいて非テキスト要素を表と見なす請求項283記載の装置。

【請求項285】 更に、非格子状配置の白輪郭を再接続する手段を備え、非テキスト要素には前記再接続手段での再接続率に基づいて表の識別子が付加される請求項283記載の装置。

【請求項286】 更に、非テキスト要素内の白輪郭の充填率を算出する算出手段を備え、充填率が高くない場合に非テキスト要素に表の識別子を付加する請求項283記載の装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明は、文字認識方法及び装置に関わるものであり、特に認識処理に先立って、画像データに基づいて画像データブロックを分類・選別する方

法や装置に関するものである。例えば、画像データがテキスト画像データであるか、中間調（あるいはグレースケール）画像、線画及びフレームなどのような非テキスト画像であるかに応じて、画像データブロックを選別して分類するものである。また、本発明は、認識処理への準備として、テキストブロックからテキストラインを識別して分割し、次にテキストライン中の個々の文字を識別してライン中の他の文字から切り出す方法や装置にも関するものである。

【0002】尚、本出願は付録のコンピュータプログラム（図35～図247参照）とともに提出される。本明細書の開示の一部には、著作権保護に関係するものが含まれている。著作権者は、特許・商標庁の特許ファイルあるいは記録なので本出願の開示文書がファクシミリ再生されることに対しては異議を唱えないが、そうでない場合にはすべての著作権を留保するものとする。

【0003】

【従来の技術】近年、テキストデータ画像を分析してテキストデータ中の個々の文字を認識し、認識された文字に対応してコンピュータが理解可能な文字コードファイルを生成することが可能になってきた。ここで得られたファイルは、ワードプロセッシングやデータプロセッシングのプログラムで操作を加えることができる。以下「文字認識システム」と呼ぶことにするこのようなシステムの利点は、テキストデータを再タイプしたり、あるいは再入力する必要がないことである。例えば、ファクシミリで送られてきた文書や、マイクロフィルムや複写機で再生された文書を文字認識して、文書中の文字や数字を文字コード（例えばアスキーコードコード）からなるコンピュータテキストファイルとして生成することにより、文書の再タイプや再入力をせずに、文書をワードプロセッシングやデータプロセッシングすることができる。

【0004】文字認識すべき文書にはさまざまな種類の画像が含まれることが多く、すべてが認識可能というわけではない。例えば、テキスト画像データの認識は現在でも可能であるが、非テキスト画像データの認識は不可能である。一般に、文字認識すべき文書には、テキスト画像データブロックと中間調画像、線画及び線などの非テキスト画像データブロックとが含まれる。また、文書には、フレームで囲まれていたり囲まれていなかったりする表や表形式のデータなども含まれる。したがって、文字認識処理に先立って、ブロック中の画像データの種類に応じて文書中の個々のブロックを分類し、画像データからテキストタイプのブロックを選別することが必要となる。

【0005】図32は典型的な文書の1ページの例を示す図である。図32において、文書ページ401は2段組のフォーマットで構成されている。ページには、タイトルに適した大きなフォントサイズのテキスト情報が含

まれるタイトルブロック402、テキストデータラインを含むテキストブロック404、テキストではないグラフィックス画像を含むグラフィックスブロック405、テキストや数字情報の表を含む表ブロック406、小さなサイズのテキストデータでグラフィックスブロックあるいは表ブロックに対する見だしが含まれる見だしブロック407が存在する。各情報ブロックはブロック内の情報の種別に応じて分類され、この分類結果に基づいてブロックが分割される。

【0006】画像データからテキスト型のブロックを検出するため、従来は、画像データ中の黒画素を隣接する1つあるいは複数の白画素にまで水平方向・垂直方向に膨張させて、水平方向・垂直方向に画素画像データを塗りつぶす手法を用いていた。このような塗りつぶし法の欠点としては、適切な塗りつぶしパラメータを選択するために、前もってテキスト型の画像データの特徴（例えばフォントサイズなど）を知らなければならないことが挙げられる。また、塗りつぶしパラメータの僅かな変更が、選別結果の大幅な変化につながるという問題もある。さらに、塗りつぶし法では、必ずしも原文書の内部構造が保存されるとは限らない。例えば、塗りつぶし法では、2段組の原文書が1段に塗りつぶされてしまうことがある。このような場合には、テキストデータの格納順序がばらばらになってしまい、原テキストの正確な再生が不可能という問題が生じる。この他、テキスト型のデータが非テキスト型のデータまで塗りつぶして、全領域をテキスト型のデータと誤って解釈してしまう場合も、塗りつぶし法ではしばしば生じる。

【0007】ブロック選別に続いて、文書中の文字ごとに文字認識処理を行い、文字に対応するコンピュータコードを得る。ここで、文字ブロックから個々の文字を得るには、2つの処理を行う。第1段の処理では、タイトルブロック202、テキストブロック204、見だしブロック207などの各テキストブロック中の個々のラインが、テキストブロック中の他のラインから分割される。一般に、ライン分割は、各ブロック中の画素密度を水平投影してライン間のギャップを識別することで行われる。すなわち、図33Aに示すように、テキストブロック404は間にギャップ412を有するテキストラインからなる。画素密度の水平投影414は、ブロック404の各ライン上の黒画素数の総和として求められる。テキストラインは投影414中で密度が非ゼロの領域に対応し、テキストライン間のギャップは投影414中の密度がゼロの領域に対応する。したがって、投影密度に基づいて、テキストラインを各ラインごとに分割することができる。

【0008】第2段の処理では、分割されたテキストライン中の個々の文字がテキストライン中の他の文字から切り出される。すなわち、図34Aに示すように、テキストライン411には個々の文字が含まれる。テキスト

法や装置に関するものである。例えば、画像データがテキスト画像データであるか、中間調（あるいはグレースケール）画像、線画及びフレームなどのような非テキスト画像であるかに応じて、画像データブロックを選別して分類するものである。また、本発明は、認識処理への準備として、テキストブロックからテキストラインを識別して分割し、次にテキストライン中の個々の文字を識別してライン中の他の文字から切り出す方法や装置にも関するものである。

【0002】尚、本出願は付録のコンピュータプログラム（図35～図247参照）とともに提出される。本明細書の開示の一部には、著作権保護に関係するものが含まれている。著作権者は、特許・商標庁の特許ファイルあるいは記録なので本出願の開示文書がファクシミリ再生されることに対しては異議を唱えないが、そうでない場合にはすべての著作権を留保するものとする。

【0003】

【従来の技術】近年、テキストデータ画像を分析してテキストデータ中の個々の文字を認識し、認識された文字に対応してコンピュータが理解可能な文字コードファイル（20）を生成することが可能になってきた。ここで得られたファイルは、ワードプロセッシングやデータプロセッシングのプログラムで操作を加えることができる。以下「文字認識システム」と呼ぶことにするこのようなシステムの利点は、テキストデータを再タイプしたり、あるいは再入力する必要がないことである。例えば、ファクシミリで送られてきた文書や、マイクロフィルムや複写機で再生された文書を文字認識して、文書中の文字や数字を文字コード（例えばアスキーコードコード）からなるコンピュータテキストファイルとして生成することにより、文書の再タイプや再入力をせずに、文書をワードプロセッシングやデータプロセッシングすることができる。

【0004】文字認識すべき文書にはさまざまな種類の画像が含まれることが多く、すべてが認識可能というわけではない。例えば、テキスト画像データの認識は現在でも可能であるが、非テキスト画像データの認識は不可能である。一般に、文字認識すべき文書には、テキスト画像データブロックと中間調画像、線画及び線などの非テキスト画像データブロック（40）とが含まれる。また、文書には、フレームで囲まれていたり囲まれていなかったりする表や表形式のデータなども含まれる。したがって、文字認識処理に先立って、ブロック中の画像データの種類に応じて文書中の個々のブロックを分類し、画像データからテキストタイプのブロックを選別することが必要となる。

【0005】図32は典型的な文書の1ページの例を示す図である。図32において、文書ページ401は2段組のフォーマットで構成されている。ページには、タイトルに適した大きなフォントサイズのテキスト情報が含

まれるタイトルブロック402、テキストデータラインを含むテキストブロック404、テキストではないグラフィックス画像を含むグラフィックスブロック405、テキストや数字情報の表を含む表ブロック406、小さなサイズのテキストデータでグラフィックスブロックあるいは表ブロックに対する見だしが含まれる見だしブロック407が存在する。各情報ブロックはブロック内の情報の種別に応じて分類され、この分類結果に基づいてブロックが分割される。

【0006】画像データからテキスト型のブロックを検出するため、従来は、画像データ中の黒画素を隣接する1つあるいは複数の白画素にまで水平方向・垂直方向に膨張させて、水平方向・垂直方向に画素画像データを塗りつぶす手法を用いていた。このような塗りつぶし法の欠点としては、適切な塗りつぶしパラメータを選択するために、前もってテキスト型の画像データ中の特徴（例えばフォントサイズなど）を知らなければならないことが挙げられる。また、塗りつぶしパラメータの僅かな変更が、選別結果の大幅な変化につながるという問題もある。さらに、塗りつぶし法では、必ずしも原文書の内部構造が保存されるとは限らない。例えば、塗りつぶし法では、2段組の原文書が1段に塗りつぶされてしまうことがある。このような場合には、テキストデータの格納順序がばらばらになってしまい、原テキストの正確な再生が不可能という問題が生じる。この他、テキスト型のデータが非テキスト型のデータまで塗りつぶして、全領域をテキスト型のデータと誤って解釈してしまう場合も、塗りつぶし法ではしばしば生じる。

【0007】ブロック選別に続いて、文書中の文字ごとに文字認識処理を行い、文字に対応するコンピュータコードを得る。ここで、文字ブロックから個々の文字を得るには、2つの処理を行う。第1段の処理では、タイトルブロック202、テキストブロック204、見だしブロック207などの各テキストブロック中の個々のラインが、テキストブロック中の他のラインから分割される。一般に、ライン分割は、各ブロック中の画素密度を水平投影してライン間のギャップを識別することで行われる。すなわち、図33Aに示すように、テキストブロック404は間にギャップ412を有するテキストラインからなる。画素密度の水平投影414は、ブロック404の各ライン上の黒画素数の総和として求められる。テキストラインは投影414中で密度が非ゼロの領域に対応し、テキストライン間のギャップは投影414中の密度がゼロの領域に対応する。したがって、投影密度に基づいて、テキストラインを各ラインごとに分割することができる。

【0008】第2段の処理では、分割されたテキストライン中の個々の文字がテキストライン中の他の文字から切り出される。すなわち、図34Aに示すように、テキストライン411には個々の文字が含まれる。テキスト



ラインにおいて各文字を他の文字から切り出すために、ラインセグメント411の各列ごとに黒画素数の垂直方向の総和を求め、画素密度の垂直投影416を得る。文字415は投影416中で密度が非ゼロの領域に対応し、文字間のギャップは投影416中で密度がゼロの領域に対応する。こうして、個々の文字がラインセグメント中の他の文字から切り出される。

【0009】このような処理においてはいくつかの問題点が存在する。例えば、文書が斜めに画像スキャナに入力され、図33Bに示すように傾斜角 $\theta_s$ で画像メモリに格納される場合が多いが、このような場合には、第1ライン418のテキストが第2ライン419のテキストとライン420で重なってしまうため、必ずしもライン分割が可能であるとは限らない。すなわち、画素密度の水平投影421は非ゼロ値のみとなり、ゼロ値が存在しないためライン間のギャップの検出が不可能になる。

【0010】このような問題点に鑑みて、テキストブロック404を図33Cに示すような複数の列422、424に分割して、各列ごとに独立に水平投影を得る手法が用いられている。すなわち、図33Cに示すように、水平投影422aは列422に対応し、水平投影424aは列424に対応するものである。各列でテキストラインが重なっていなければ、各列中のテキストラインの検出が可能となる。図33Cでは2列に分割する例を示しているが、一般には5列から10列に分割し、テストで最大傾斜角 $\theta_s$ までの傾きを有していても、ブロック中の他のラインから個々のラインの分割が可能であるようにしている。

【0011】しかし、各列ごとに水平画素投影を得なければならない、またこのようにして得た各水平画素投影も個々に処理しなければならないため、ライン分割処理がきわめて時間のかかる処理となる。また、最大傾斜角 $\theta_s$ の文書にまで対応しなければならないため、傾斜角が小さくて1列あるいは数列への分割で十分である多くの文書に対しても、すべての列を処理しなければならない、この点でも時間のかかる処理となっている。上記のような2段階処理のうち1つの問題は、個々の文字をラインセグメント中の他の文字から切り出す第2段の処理において生じる。

【0012】すなわち、文字間に垂直な空間が存在する場合には、図34Aに示したような処理で良好な結果が得られるが、文字が垂直方向に重なっていた場合や2つあるいは複数の文字が接触している場合には、文字分割処理を適切に行うことができない。イタリックフォントの場合や、複数回の複写処理やファクシミリ送信によって画像品質が劣化している場合など、このような状況に陥る可能性は高い。例えば、図34Bに示すように、イタリックテキストでは単語「Satisfy」中の文字「f」と「y」とが垂直方向に重なっており、画素密度の垂直投影425が文字間でゼロにならないことがあ

る。そのため、文字「f」と「y」とを分離することはできない。また、文字「t」と「i」も接触しているため、同様にこれら2つの文字の分離も不可能である。

【0013】

【発明が解決しようとしている課題】本発明は、前記従来の欠点を除去し、高速かつ正確に文書上の文字を認識してテキストファイルを作成する文字認識方法及び装置を提供する。また、高速かつ正確に文書上のテキストと非テキストとを選別して、テキストブロックを割り出す方法及び装置を提供する。

【0014】また、高速かつ正確に文書上の非テキストを分類する方法及び装置を提供する。また、高速かつ正確にテキストブロックからテキストラインを分割する方法及び装置を提供する。また、高速かつ正確に傾いた文書のテキストブロックからテキストラインを分割する方法及び装置を提供する。

【0015】また、高速かつ正確にテキストラインから文字を切り出す方法及び装置を提供する。また、不適切に切り出された文字の再切り出しが可能な方法及び装置を提供する。

【0016】

【課題を解決するための手段】この課題を解決するために、本発明の文字認識方法及び装置は以下の構成を含む。画素画像データが入力され、2値の画素画像データでない場合は2値の画素画像データに変換される。

【0017】画素画像データのブロックが、画素画像データ内で連結要素の輪郭を追跡し、連結要素がテキスト要素を含むか非テキスト要素を含むかを連結要素のサイズに基づいて決定し、隣接するテキスト要素の近接状態に基づいてテキスト要素を選択的に連結してテキストラインを形成し、隣接するテキストラインの近接状態とテキストライン間の非テキスト要素の位置とに基づいてテキストラインを選択的に連結することにより、選別される。階層的木構造が連結要素に基づいて形成される。

【0018】テキストブロックは、段の全域での画素密度の水平投影に基づいてテキストブロックを少なくとも1つのラインに分割することで、テキストラインに分割され、非接触で重なっていない文字を切り出す第1の切り出しと接触文字間を切り離す第2の切り出しとの2回の切り出しで、文字が分割されたラインから切り出される。切り出された文字は認識され、認識に基づいて文字コードが割り当てられる。文字コードは階層的木構造による確定された順でコンピュータのテキストファイル内に格納される。

【0019】必要ならば、非テキスト要素は階層的木構造により確定された順に対応して格納された文字コードの間に散在してもよい。画素画像データは例えば画像圧縮や画質向上等の前処理をされ、認識された文字は例えば文脈チェック等の後処理をされてもよい。非テキスト要素の特徴に基づいて、非テキスト要素に識別子が付与

される。例えば、非テキスト要素の内部で白輪郭の追跡が行なわれ、非格子状配置の白輪郭が再連結されて、白輪郭の充填率が計算されて、白輪郭の数、非格子状配置の白輪郭の再連結率、あるいは白輪郭の充填率等に基づいて、非テキスト要素には表の識別子が付与される。

【0020】非接触で重なっていない文字の切り出しは、ラインセグメントへのとびとびの処理で達成され、接触文字の切り離しは、接触文字間の空間に関する情報が既知か否かに対応して達成される。空間に関する情報が既知の場合は、空間の統計値に基づいて切り出しが達成される。空間に関する情報が無い場合、接触文字間の切り離しは、回転された投影によって決められた角度と位置で接触文字を斜めに切り離すように、画素密度の回転投影に対応して達成される。不適切に切り出された文字は再接続される。

【0021】

【実施例】本発明は、複写機、ファクシミリ、ビデオカメラあるいはスチルビデオカメラ、レーザビームプリンタなどの画像処理装置あるいは画像再生装置などのような、文字認識処理が望まれるさまざまな装置において実施可能である。このような装置においては、文字画像を有する画像を、文字画像が文字認識されるように処理あるいは再生する。この際、認識した文字画像を標準文字セットあるいはフォントに替えて、原文字画像ではなく標準文字を再送あるいは再生することもできる。また、本発明は、汎用コンピュータや、パーソナルコンピュータ、ワードプロセッシングあるいはデータプロセッシング機器などのオフィス機器や、複数のビジネス機器を単一の統合パッケージにまとめた統合オフィス自動機器などにおいても実施可能である。

【0022】図1は、スキャン、ファクシミリ、情報送受信、情報処理（ワードプロセッシングやデータプロセッシングなどの処理をも含む）などの機能を有する統合オフィス自動機器として、本発明の一実施例を示すブロック図である。図1に示す装置において、画像の入力は、ファクシミリ送信による入力、原文書のスキャンによる入力、モデムを介して離れた地点からの入力などで行われる。本実施例によれば、画像中の文字を認識し、認識文字のコンピュータテキストファイルを生成し、装置内のワードプロセッシング、スプレッドシート処理ならびに他の情報処理機能を利用してテキストファイルの修正を行うことができる。この処理に続いて、修正されたテキストファイル（あるいは無修正のテキストファイル）の再送や、音声合成技術を用いてテキストファイルのスピーカや通常の音声電話への「話しかけ」などによる出力が行われる。

【0023】図1において、プログラム可能なマイクロプロセッサなどの中央処理ユニット（CPU）10はバス11に接続されている。また、画像を画素ごとにスキャンして画像メモリ（例えば、以下で示すRAM20）

に蓄積するスキャナ12、デジタルデータを電話線15aを通してアナログ形式で送受信するモデム14、画像データを電話線15bを通して送受信するファクシミリ16（不図示の電話を含むこともある）なども、バス11に接続される。なお、電話線15aと15bとは同一の線であって、不図示のネットワーク制御ユニットで制御されるものでも良い。さらに、バス11には、CPU10によって実行される1つあるいは複数のコンピュータプログラムを蓄えるリードオンリメモリ（ROM）17、認識処理において入力文字と比較する文字の辞書を保持している文字辞書19、入力画像データ、処理画像データならびに画像の構造などの情報を蓄えるランダムアクセスメモリ（RAM）20、文字認識処理において文字の認識結果を出力する出力装置21（ディスクメモリ、スピーカあるいは音声電話線インタフェースを有する音声合成装置など）、装置によって処理された画像を表示するプリンタ/ディスプレイ22、オペレータが装置を操作するためのキーボード24なども接続される。

【0024】なお、ここでは、バス11に接続された装置が協調して統合オフィス自動機器を構成しているが、これらの装置のうちのいくつかあるいはすべてが単独で用いられることも可能なことは明らかであろう。スキャナ12、モデム14及びファクシミリ16は、画像データを装置に入力するための種々の入力手段である。スキャナ12では、原画像はラインごとと画素ごとにスキャンされ、CPU10の制御のもとで画像データの画素はRAM20中の画像メモリにビットマップ形式で蓄えられる。モデム14では、電話線15aを介してアナログ形式で受信した画像データがデジタル画素形式に変換され、デジタル画素データはRAM20中の画像メモリに蓄えられる。ファクシミリ16では、電話線15bを介して修正ハフマンランレンクス符号などの符号形式で受信した圧縮画像データを、ファクシミリ16において従来手法でデジタル画像の画素データに復号して、CPU10によって画像データの画素はRAM20中の画像メモリにビットマップ形式で蓄えられる。ここで、他の画像入力手段を用いることはもちろん可能で、ディスクメモリなどの大容量の蓄積メディアから画像データを取り出したり、ビデオカメラあるいはスチルビデオカメラから入力することもできる。

【0025】ファクシミリ16や出力装置21は、文字認識された画像データを装置から出力するための種々の手段である。ファクシミリ16では、本実施例に基づき認識処理された文字画像が標準文字セットあるいはフォントに変換され、装置から送信される。これにより、例えば、文字画像を含む画像を受信して、文字画像を文字認識し、再送に先立って認識文字を標準文字フォントに変換することで、劣化画像の品質の改善を図ることができる。

【0026】モデム14や出力装置21は、画像データ



で認識された文字をASCIIコードなどで出力あるいは蓄積するための種々の手段である。すなわち、文字は装置（ディスクメモリなど）に蓄えられたり、モデム14を介して離れた地点に転送するために出力される。この際、ASCIIコードをファクシミリ互換のフォーマットに変換するなどの文字変換手段を設ければ、ファクシミリ16を用いることなくモデム14を介して離れた地点のファクシミリ機に転送を行うことができる。

【0027】プリンタ/ディスプレイ22は、本実施例に基づく文字認識処理の流れの監視、ならびに文字認識の各ステップの永久記録の出力および表示を行うための手段である。キーボード24は、図1の装置の動作に対するオペレータの制御を可能にしている。図2は、本実施例の文字認識処理のフローチャートである。図2に示した処理ステップは、プログラムROM17に蓄えられているコンピュータプログラムに基づいてCPU10で実行される。

【0028】ステップS201において、画素単位の画像データ（以下画素画像データ）は装置に入力され、RAM17に蓄えられる。画像データは画像を画素ごとに表現したものである。この際、画素データは2値画素データ、すなわち黒ならびに白画像データであることが好ましい。なお、画像データは、複数のグレイ階調値の1つで各画素が表現される中間調画像であっても、画素色を表す複数ビットワードで各画素が表現されるカラー画像データであっても良い。このような場合、すなわち画素データが2値画素データでない場合には、非2値画素データを2値画素データに変換するためのしきい値処理をRAM20に蓄積する前に行う必要がある。

【0029】ステップS201で入力された画素画像データは、左上隅から右下隅に読み進められるようなポートレート画像であることが好ましい。画像がポートレート画像でない場合、例えばランドスケープ画像である場合には、画素画像データをポートレート型の画像に変換する必要がある。この変換処理は、オペレータがキーボード24から画像データの変換を指示することで行うことができる。

【0030】ステップS202では、画像データの前処理が行われる。一般に、前処理フィルターを用いて画像データの画質を向上し、劣化した文字や画像を改善する。画質向上手法として適切な手法は、1991年10月4日出願の米国出願番号07/771,220に示されている。尚、ステップS202では、精度の劣化は避けられないが、画素画像データ中の複数の画素を削除したり圧縮したりすることで、認識処理の高速化を図ることも可能である。例えば、 $m \times n$ 画素ブロックの画素の平均値を求め、この平均値で $m \times n$ 画素ブロックを代表させることも可能である。

【0031】ステップS203では、ブロック選択を行って各セグメント画像中の画像データの型を調べ、プロ

ック中の情報がテキスト情報か、グラフィックス情報か、線画情報か、画像情報かなどを識別する。また、ブロック選択ステップS203では、画像中の各部分を以下のステップS212で示すような適切な順序で再生することができるように、画像の階層的木構造をも生成する。この階層的木構造には、例えば、2段組の画像データにおいて第1段のテキストから第2段のテキストに飛んで読み進めることのないように、第1段のテキストの後に第2段のテキストを再生できるような情報が含まれる。なお、ステップS203におけるブロック選択処理については、以下でより詳細に説明する。

【0032】RAM20に蓄えられた画像から第1の情報ブロックが選択され、ブロック選択ステップS203で判別されたブロック識別子に応じて、ステップS204において選択されたブロックがテキストブロックであるかどうか調べられる。第1ブロックがテキストブロックでない場合には、処理をステップS205に進め、次のブロックを選択してステップS204に戻る。

【0033】ステップS204においてブロックがテキストブロックである場合には、処理をステップS206に進め、テキストブロックに対してライン分割処理を行う。ライン分割処理では、テキストブロック中の個々のテキストラインをテキストブロック中の他のテキストラインから分離して、以下に詳細に説明するように分離したラインごとに順々に処理を行う。ステップS207では、ライン中の各文字をライン中の他の文字から切り出し、以下に詳細に説明するように個々の文字に対して文字認識処理を行う。

【0034】ステップS208では、文字ごとに各文字の認識処理が行われ、既知の手法で文字辞書19に蓄えられている標準文字と各文字との比較処理が行われる。文字辞書19中の索引は一般には1つの文字に対応するが、分離が難しいような文字列（例えば、「f i」）や分離されやすい1文字（例えば「j」）に対しては他の索引が用意されることもある。すなわち、分離が難しいような接触文字ペアを辞書の索引とすることも、分離されやすいような1文字中の部分を辞書の索引とすることもある。比較処理に基づいて文字画像に対する識別子が選択され、選択された文字識別子はRAM20に蓄えられたり、出力装置21に出力される。また、識別された文字をプリンタ/ディスプレイ22に表示することもできる。

【0035】ステップS209では、テキストブロックに対する処理が終了したかどうか調べられる。処理が終了していなければ、処理をステップS206（あるいは適宜ステップS207）に戻し、ライン分割処理、文字切り出し処理及び文字認識処理とを繰り返す。ブロック処理が終了していれば、処理をステップS210に進め、1ページ分の処理を終了したかどうか調べる。1ページ分の処理が終了していなければ、処理をス

テップS205に戻し、ページ中の次ブロックを選択して処理を続ける。

【0036】1ページ分の処理が終了していれば、処理をステップS210からステップS211に進め、後処理を実行する。後処理では、文脈チェックやスペルチェックなどの処理を行い、文字認識ステップS208で認識された文字識別子を、ステップS208のような個々の文字ごとではなく文字の周囲の文脈に基づいて、全体的な見地から修正する処理を行う。ステップS212では、ブロック選択ステップS203で定義された階層的木構造に基づいて画像データが再生される。ページ再生では画像データが適切な順序で配置される。例えば、脚注はメインテキストから分離され、段は他の段と混ざることなく順番に配置され、グラフィックスや線画はページ中の認識文字テキストの適当な位置に挿入される。このようなグラフィックス画像や線画の見だしも図に隣接して挿入される。この際、他の規則を用いることもでき、例えばページの物理的再生ではなく、ページからテキストのみを抽出するという規則を用いることもできる。なお、このようなページ再生の規則は、装置の情報処理機能を用いてユーザが定義できる。

【0037】ステップS213では、再生されたページが出力装置21などに蓄えられる。ここでは、ROM17に蓄えられており、CPU10で実行される他の応用プログラムを用いて、スプレッドシートやワードプロセッシング処理などの情報処理を行うことができる。そして、処理された情報（あるいは適宜処理していない情報）は、ファクシミリ16やモデム14などを用いて、またはコンピュータテキストファイルを「しゃべる」音声合成装置を用いた通常の音声電話を介してなどのさまざまな手段を用いて再送される。

【0038】図3は、図2のステップS203のブロック選別処理を説明するための一般化した処理流れ図である。ここで、ブロック選別処理は文字認識システムとは別に利用可能であることに注意されたい。すなわち、画像再生装置においてブロック選別処理を用いて、ある種のブロックに対しては画像再生の第1の手法を施し、別の種類のブロックに対しては画像再生の第2の手法を施すといったことも可能である。

【0039】ブロック選別の処理速度の向上を望む場合には、ステップS300で画像データを縮小する処理を行う。画像データの縮小処理を行った場合には、ブロック選別処理は縮小画像に対して行われる。なお、この際、図2の残りの文字認識処理（ステップS204からステップS213）に影響を及ぼさないように、ブロック選別処理終了時には、縮小前の画像データに対して選択ブロックが割り当てられる。画像データ縮小処理は、 $m \times m$ 画素ブロック中の黒画素の連結性を調べながら行われる。例えば、 $3 \times 3$ 画素ブロック中に連結された黒画素が2個ある場合には、 $3 \times 3$ 画素ブロックを1つの

黒画素に縮小する。逆に、 $3 \times 3$ 画素ブロック中に連結された黒画素が2個ない場合には、 $3 \times 3$ ブロックを1つの白画素に縮小する。

【0040】ステップS301では、画素画像を解析して連結された要素を検出し、連結要素をサイズや他の連結要素との相対的な位置に応じて分類する処理を行う。ここで、連結要素は白画素に完全に囲まれた黒画素のグループである。すなわち、連結要素は、少なくとも1つの白画素で他の黒画素グループと完全に分離されている黒画素グループである。図4を参照して以下で詳細に説明するように、ステップS301では、連結要素を検出して、連結要素から得たサイズ情報やいくつかの統計的な値に基づいて各連結要素の分類を行う。

【0041】まず、以下で詳細に説明するように、各連結要素をテキスト要素か非テキスト要素かに分類する。さらに、非テキスト部については、フレームデータであるか、中間調画像であるか、線画であるか、表あるいはテキストデータを表のように構成したものであるか、あるいは未知の要素で分類不可能であるかなどを解析処理して決定する。また、各連結要素ごとに階層的木構造を構成して、連結要素の構造的な記述データを求め、ステップS212で説明したようにデジタルデータの再生を容易にする。

【0042】ステップS302では、水平方向に近い位置関係にあり、ギャップラインマーカを横切らないような連結要素をラインとしてグループ化する。この際、ステップS301で作成した木構造を用いて、テキスト要素と非テキスト要素とが不適切にグループ化されないようにする。また、ステップS302では、段間で垂直方向に伸びるギャップや非テキスト要素の垂直方向に伸びる境界を検出して、テキスト要素を段ごとにまとめる。ここで得られた段構造は階層的木構造に組み込まれ、適宜木構造の更新が行われる。

【0043】ステップS303では、ステップS302でグループ化されたラインの内、垂直方向に近接するラインを垂直方向にグループ化して、ブロックを形成する。ここで、非テキスト要素は垂直方向にソートされ、画像ページの境界として用いられる。なお、2つの非テキスト要素の間に位置するテキストライン要素は、他のテキストライン要素とは別に処理される。また、ステップS303では、ステップS301で分類できなかった非テキスト要素が、大きなフォントサイズのタイトルであるかどうかを判別する。タイトルであると判別されると、その部分に「タイトル」属性を付与し、木構造を更新する。ここで得られたタイトルは、ステップS212に基づくページの再生の際に有用な情報となりうる。

【0044】図4A～図4Cは、画素画像データの連結要素をどのように検出して、どのようにこれらの連結要素を分類するかを示す詳細なフローチャートである。図4A～図4Cに示される処理ステップは、ROM17に

保持されているプログラムステップに基づいてCPU10で実行される。ステップS401では、輪郭追跡を行い画素画像データの連結要素を検出する。輪郭追跡処理は図5Aに示すように画像データをスキャンすることで行われる。スキャン処理は矢印Aで示されるように画像の右下部から左に行われ、画像の右側の端に達したときに上に進むように行われる。なお、この際、逆方向すなわち左上から右下にスキャンすることもできる。スキャン処理中に黒画素を検出すると、31で示すような星形パターンの順序で隣接画素を調べ、黒画素の隣接画素もまた黒画素であるかどうかを判別する。星形パターン31には8個の番号付けされたベクトルが中心点から伸びているため、このような輪郭追跡を以下「8方向」追跡と呼ぶことにする。黒画素が隣接して存在する場合には、画像の外側輪郭を追跡し終えるまで上述の処理を繰り返す。

【0045】すなわち、図5Bに示すように、矢印A方向へのスキャン処理によって、まず文字「Q」の端に対応する点32が検出される。次いで、星形パターン31に基づいて隣接画素が判別され、文字「Q」の外側輪郭が追跡される。この際、閉輪郭の内部の輪郭追跡処理は行わない。連結要素を1つ検出して、8方向追跡で輪郭の追跡を終えると、次の黒画素を検出するまでスキャン処理を続ける。すなわち、完全な黒領域である物体34の8方向追跡を行う。また、手書き単語「non-text」の非テキスト物体35を、単語「word」を構成する個々の文字のテキスト物体36中の個々の文字と同様に追跡する。図5Aに示すスキャン処理は、画素データ中のすべての連結要素を検出し、8方向追跡して輪郭を検出するまで行われる。

【0046】次いで処理をステップS402に進め、各連結要素を矩形で囲む処理を行う。具体的には、各連結要素の周りを包囲する最も小さな矩形を描く。すなわち、図5Bに示すように、物体32の周りには矩形37が、物体34の周りには矩形39が、物体35の周りには矩形40が、テキスト物体36a、36b、36c、36dの周りには矩形41a、41b、41c、41dが描かれる。

【0047】ステップS403では、各矩形要素に対して木構造中の位置が割り当てられる。ほとんどの場合、ステップS403で得られる木構造では、画素画像中の各物体は木の根に直接つながる。これは、連結要素の外側輪郭のみを追跡しており、閉輪郭の内部は追跡していないためである。すなわち、図5Cに示すように、連結要素32に対応する矩形37はページの根に直接つながる。これに対し、非テキスト物体35に対応する矩形40や、テキスト物体36a、36bに対応する矩形41a、41bなどのように、矩形が他の矩形の内部に完全に位置するような連結要素は、囲っている連結要素（この場合には要素34）の子ノードとして位置付けられ

る。また、要素34などのように少なくとも1つの子を有する連結要素は、それ自身を「主な子ノード」として位置付ける。すなわち、図5Cに示すように、要素39が「主な子ノード」として、要素39の他の子ノード40、41a、41bとともに位置付けられる。

【0048】ステップS404では、木構造の第1レベルの各連結要素をテキスト要素あるいは非テキスト要素に分類する。この分類処理は2つのステップで行われる。第1ステップでは、連結要素の矩形と所定のしきい値サイズとの比較を行う。連結要素を囲む矩形の高さが、想定される最大フォントサイズに対応する所定の第1しきい値以上であれば、あるいは連結要素を囲む矩形の幅が、ページ幅を経験的に決められた一定値で割った値以上であれば（「5」で良好な結果が得られることがわかっている）、連結要素を非テキスト要素と分類し、要素に「非テキスト」の属性を付与する。

【0049】第2ステップでは、残りすべての要素すなわち非テキストと分類されなかった要素と、残りすべての連結要素のサイズの集合に基づいて適当に求められるしきい値との比較を行う。具体的には、非テキスト要素として分類されなかったすべての矩形の平均の高さを求め、この平均高にスカラー値を掛け合わせたものを（「2」を用いると好都合である）適当に求められるしきい値とする。そして、適当に求めたしきい値以上に大きな要素はすべて非テキストであると考え、非テキスト要素として分類する。また、適当に求めたしきい値以下の小さな要素はすべてテキスト要素であると考え、このようにして要素の分類を行い、適切な属性が付与される。

【0050】なお、図4A～図4Cの残りを参照して以下で詳細に説明するように、ここで求めた分類結果は以下で改善される。木構造の第1レベルの各要素をテキスト要素あるいは非テキスト要素として分類すると、テキスト要素の主な子ノードを含むすべての子ノードをテキスト要素として分類する。これに対して、非テキスト要素の主な子ノードは非テキスト要素として分類するが、非テキスト要素の他のすべての子ノードはテキスト要素と分類する。

【0051】ステップS405において、最初の要素を選択する。要素がテキスト要素であれば（ステップS406）、ステップS407に処理を進め次の要素を選択する。非テキスト要素を選択して処理をステップS408に進めるまで、ステップS406とS407の処理を繰り返す。ステップS408では、非テキスト要素に子ノードが存在するかどうか調べられる。例えば、図5Cに示されるように、非テキスト要素39には、非テキストの主な子ノード39とテキストの子ノード40、41a、41bとが存在する。

【0052】ステップS408において子ノードが存在する場合には、処理をステップS409に進め、要素に

フィルタ処理が施され、要素が中間調（あるいはグレイ階調）要素であるかどうかを判別される。中間調フィルタ処理では要素の子ノードを調べ、要素の子ノードのうち「雑音サイズ」以下であるようなサイズのものを求める。ここで「雑音サイズ」の要素とは、高さが画像データにおいて想定される最小のフォントサイズ以下であるような要素のことである。雑音サイズ以下のサイズの子ノード数が全体の子ノード数の半分以上であれば、要素は中間調画像として判断される。そこで、ステップS410で処理をステップS411に進め、「中間調」属性を要素に付与する。

【0053】次いでステップS412で、中間調画像中のテキスト要素をチェックする。具体的には、木構造中の中間調画像の子ノードのうちテキストサイズの子ノードを修正して、テキストサイズの要素を中間調画像の子ノードとしてではなく、中間調画像と同一レベルのノードに配置する。これにより、中間調画像中のテキストサイズの要素に対しても、適切だとすれば文字認識処理を行うことができる。この処理を終えると処理をステップS407に戻し、次の要素を選択して処理を行う。

【0054】ステップS409の中間調フィルタ処理において要素が中間調画像でないと判別された場合には、処理をステップS410からステップS413に進め、さらなる処理を施すために要素の主な子ノードを選別する。そして、処理をステップS414に進める。ステップS408において非テキスト要素が子ノードをもたないと判別される場合、あるいはステップS413においてさらなる処理を施すために主な子ノードが選択されると、ステップS414で当該要素に対してフレームフィルタ処理が施される。フレームフィルタ処理は、当該要素がフレームであるかどうかを調べ、当該要素を囲むような矩形とほぼ同一の幅/高さを有する平行水平線と平行垂直線とを検出するための処理である。具体的には、画素中の行ごとに当該要素中の連結要素内部の最大長を求めて、連結要素が調べられる。

【0055】すなわち、図6Aに示されるように、非テキスト要素42には連結要素43が含まれるが、その輪郭は8方向追跡によって44で示すように追跡処理される。行「I」における連結要素内部の最大長は、輪郭の左端45aから右端45bまでの距離 $x_1$ となる。また、行「J」では、連結要素内部の長さとして、連結要素の境界上の点46aと46bの距離、ならびに点47aと47bの距離との2つが存在する。ここで、点46aと46bの距離の方が点47aと47bの距離よりも長いため、距離 $x_1$ が行「J」における連結要素内部の最大長となる。

【0056】非テキスト要素42中のn個の行ごとに距離「x」を求め、以下の不等式を満たすかどうかを調べ、非テキスト要素がフレームであるかどうかを判別する。

【0057】

【数1】

$$\sum_{k=1}^N \frac{(X_k - W)^2}{N} < \text{threshold} \quad (\text{しきい値})$$

ここで、 $x_k$ は（上述のように）連結要素内部のk行目の最大長、Wは矩形要素42の幅、Nは行数、しきい値はフレームが画像データ中で傾いていてもフレーム検出が可能となるように前もって計算した値である。ここで、1度の傾き角を許容するには、しきい値として $\sin(1^\circ)$ を1倍したものに、ステップS404で計算したテキストの平均の高さをオフセットとして加えたものを用いると良い。

【0058】上記の不等式が満たされれば、要素はフレーム要素であると判断され、ステップS415からステップS416に処理を進め、「フレーム」属性を要素に付与する。ここで、複数の属性が各要素に付与されることに注意されたい。すなわち、「フレーム-表」、「フレーム-中間調」などのようにフレームの属性が付与されることもある。

【0059】ステップS416に続いて、フレーム要素中に表データあるいは表のように組織されたデータが存在するかどうかを調べる処理を行う。そこで、ステップS417において、連結要素の内部を調べて白輪郭を求める。白輪郭は、黒画素ではなく白画素に着目するという点を除けば、上のステップS401で検出した輪郭と同一のものである。すなわち、図7Aに示されるように、非テキスト要素の内部を非テキスト要素の右下から左上へ矢印Bの方向に沿ってスキャンする。このスキャン処理によって第1の白画素が検出されると、星形パターン51の順序で白画素の隣接画素がチェックされる。星形パターン51には1から4までの数が付与されたベクトルがある。このため、このような処理に基づく白輪郭追跡処理のことを以下「4方向」白画素追跡と称する。

【0060】黒画素に囲まれた白輪郭すべてを追跡し終えるまで、4方向の白画素追跡処理を繰り返す。例えば、白輪郭追跡処理によって、黒画素セグメント52、53、54、55ならびに内部に存在する56のような他の黒画素で構成される内部輪郭の画素が追跡される。このようにして白輪郭を位置付けると、矢印B方向へのスキャン処理を、非テキスト物体に囲まれたすべての白輪郭を追跡するまで繰り返す。

【0061】ステップS418では、非テキスト要素の密度を計算する。密度は連結要素中の黒画素数を計数して、黒画素数を矩形中の全画素数で割ることで計算される。ステップS419では、非テキスト要素中で検出された白輪郭数の数をチェックする。白輪郭の数が4以上であれば、非テキスト画像が表またはテキストブロックを表のように並べたものである可能性が高い。

【0062】そこで、ステップS420において、白輪郭の充填率を求める。白輪郭の充填率は、白輪郭が非テキスト画像中の領域を占める割合のことである。例えば、図7Aに示されるように、白輪郭充填率には、57や59のような斜線領域で完全に空の白領域と、60や61のような領域で中に黒画素を含むような白領域とが含まれる。この充填率が高いと、非テキスト画像が表またはテキストデータを表のように並べたものである可能性が高い。そこで、ステップS421において充填率を10  
チェックする。この際、高い充填率であれば、非テキスト画像は表またはテキストデータを表のように並べたものであると考えられる。

【0063】この処理の信頼性を向上させるために、白輪郭が水平ならびに垂直方向に格子状の構造をなしているかどうかを調べる。すなわち、ステップS422において、少なくとも2つの白輪郭の境界線が水平方向ならびに垂直方向に一致しないような非格子状の白輪郭を再接続する。例えば、図7Aに示されるように、白輪郭59の左境界線62及び右境界線63は、白輪郭60の左境界線64及び右境界線65と垂直方向に一致する。そのため、これらの白輪郭は格子状に配置されており、白輪郭の再接続は行わない。同様に、白輪郭59の上部境界線66及び下部境界線67は、白輪郭70の上部境界線68及び下部境界線69と水平方向に一致する。そのため、これらの白輪郭は格子状に配置されており、これらの白輪郭の再接続は行わない。

【0064】図7Aから図7Dまでは、白輪郭を再接続する場合を説明するための図である。図7Bは非テキスト要素71を示しており、これはステップS201で示した中間調画像の2値画像へのしきい値処理などによ20  
って得られるものである。非テキスト画像71には、黒領域72と白領域74、75、76、77、78、79が存在する。ここで、これらの白領域の充填率は十分高く、ステップS421において処理は再接続ステップS422に進められたものとする。

【0065】まず、図7Cに示されるように、白輪郭75の左境界線及び右境界線と、白輪郭76の左境界線及び右境界線とを比較する。すると、これらの境界線は一致しないため、白輪郭75と白輪郭76とは再接続されて図7Cの接続白輪郭76'が生成される。次いで、図7Dに示されるように、白輪郭77の上部境界線及び下部境界線と、白輪郭79の上部境界線及び下部境界線とを比較する。すると、これらの境界線は一致しないため、白輪郭77と白輪郭79とは1つの接続白輪郭77'として結合される。再接続される輪郭がなくなるまで、これらの処理を水平方向に垂直方向に繰り返す。

【0066】すなわち、上述のように、表をなす白輪郭は再接続されにくいのに対し、中間調画像や線画などの表でないような白輪郭は再接続されやすい。そこで、ステップS423において接続率を調べる。ここで、再接

続率が高ければ、あるいは再接続処理後に残った白輪郭の数が4より少なければ、処理をステップS428に進め、以下で詳述するように、非テキスト要素に「中間調画像」もしくは「線画」の属性を付与する。ステップS423において再接続率が高くなく、且つ少なくとも4つの白輪郭が残っていれば、処理をステップS424に進め、非テキスト画像に「表」の属性を付与する。

【0067】ステップS425では、新たに属性が付与された表の内部を調べ、8方向追跡処理で連結要素を検出し分類する。ステップS426では、この新たな内部の連結要素に基づいて階層的木構造を更新する処理を行う。続くステップS427では、ステップS402からステップS404で示したような手法でもって、内部連結要素をテキスト要素もしくは非テキスト要素に再分類し、適当な属性を付与する。これらの処理を終えると、処理をステップS407に戻し次の要素が選択される。

【0068】ステップS421とS423に戻って、ステップS421で充填率が高くない場合、あるいはステップS423で再接続率が高い場合には、非テキストフレーム要素は中間調画像もしくは線画である可能性が高いと考えられる。ここで、要素を中間調画像として分類するか、あるいは線画として分類するかの決定は、要素中の黒画素の平均水平ラン長、要素中の白画素の平均水平ラン長、白画素と黒画素との比率、及び密度に基づいて行われる。一般に、暗めの画像は中間調画像であると考えられ、明めの画像は線画であると考えられる。

【0069】具体的には、白画素の平均水平ラン長がほぼゼロ（すなわち、全体が暗めあるいはまだらな画像）であって、ステップS418で求めた密度によって要素が白っぽいというよりもむしろ黒っぽい場合（すなわち、密度が1/2程度の第1しきい値より高い）には、フレーム要素は中間調画像であると判別される。密度が第1しきい値以下である場合には、要素は線画として判別される。

【0070】白画素の平均水平ラン長がほぼゼロではなく、黒画素の平均水平ラン長より長い場合には、フレーム要素は線画であると判別される。しかし、白画素の平均水平ラン長が黒画素の平均ラン長以下（すなわち、暗めの画像）である場合には、さらなる判別処理が必要となる。具体的には、黒画素数が白画素数よりもかなり少ない（すなわち、黒画素数を白画素数で割った値が約2の第2しきい値より大である）場合には、フレーム要素を中間調要素として判別する。これに対し、黒画素数を白画素数で割った値が第2しきい値以下であっても、ステップS418で求めた密度が第1しきい値より高ければ、フレーム要素を中間調画像として判別する。これ以外であれば、フレーム要素は線画として判別される。

【0071】ステップS428においてフレーム要素が線画として判別されれば、処理をステップS429に進めて「線画」属性を付与し、ステップS430ですべて

の子ノードを削除する。なお、要素が線画として判別されると、線画要素のいかなるブロックも文字認識のために選択されることはない。そして、処理をステップS407に戻し次の要素を選択する。

【0072】一方、ステップS428においてフレーム要素が線画として判別されなかった場合には、処理をステップS431に進めて「中間調」属性を付与し、ステップS432でフレーム中間調要素のテキストサイズの子ノードを削除する。ここで、テキストサイズは、ステップ404において述べたように平均の要素の高さとして求められる。また、テキストサイズより大きなすべての子ノードは、フレーム中間調要素の子ノードのままとする。これらの処理を終えると、処理をステップS407に戻し次の要素を選択する。

【0073】ステップS419に戻り、白輪郭の数が4より小さい場合には、フレーム要素は表であるとは判別されない。そこで、処理をステップS433に進め、ステップS418で求めた密度と約0.5のしきい値とを比較する。ここで、しきい値は、フレーム内のテキスト要素や線画が占める面積は画素の半分以下であることを鑑みて選択されている。

【0074】密度がしきい値より小さければ、処理をステップS434に進め、上述のようにフレーム要素の内部構造を作成する。すなわち、処理をステップ401に戻し、フレーム要素の内部構造に対する処理を行う。ステップS433において、密度が所定のしきい値以上であった場合には、処理をステップS442に進め、フレーム要素を線画、中間調画像あるいは分類不可能（すなわち、フレームが「未知」）のどれかに分類する。

【0075】ステップS415に戻り、ステップS414におけるフレームフィルタ処理で非テキスト要素中にフレームが検出されなかった場合には、処理をステップS435に進め、非テキスト要素中に線が含まれるかどうかを調べる。線は、テキスト境界を明確にするのに適した非テキスト要素である。この際、このような線に囲まれたテキストは線のそばに位置することが多いため、テキストが線に接するということもありうる。そのため、テキストが接していても接していなくても線を検出できるように線の検出処理を行う。

【0076】テキストが接していない線を検出するには、非テキスト要素の長さ方向のヒストグラムを求める。すると、図6Bに示されるように、線のヒストグラム48はほぼ均一な分布となり、ヒストグラムの高さは線幅にほぼ等しくなる。また、線幅は、非テキスト要素の幅「W」にほぼ等しい。ここで、線幅と非テキスト要素の幅との差は、画素画像を生成する際の原文書の傾き角 $\theta_s$ によるものである。そこで、非テキスト要素が線を含むかどうかを調べるために、ヒストグラム中の各セルkの高さ49を非テキスト要素の幅Wと比較する。すなわち、これらの値間の実効値誤差としきい値とを次式

のように比較する。

【0077】

【数2】

$$\sum_{k=1}^N \frac{(\text{cell}_k - W)^2}{N} < \text{threshold} \quad (\text{しきい値})$$

ここで、しきい値は、非テキスト要素中の線の傾き角 $\theta_s$ を許容できるように設定される。例えば1°の傾き角であれば、しきい値として

【0078】

【数3】

$$\sum_{k=1}^N \left\{ \frac{k \sin(1^\circ)}{N} \right\}^2$$

を用いると良好な結果が得られる。上の不等式によってテキスト要素が接していない線が検出されなかった場合には、テキスト要素が接している線を含む要素であるかどうかを調べる処理を行う。テキスト要素に接している線が非テキスト要素中に含まれるかどうかを判別するために、要素の境界に沿って線が長く伸びているかどうかを調べる。すなわち、要素を通して線が長く伸びていれば、図6Cに示すように要素を囲む矩形の境界は線にきわめて近いところに位置する。そこで、矩形の境界近くの第1番目の黒画素位置の均一性を、境界からの距離の2乗の和を求めることで判断する。すなわち、次式の不等式を満たすかどうかを調べる（図6C参照）。

【0079】

【数4】

$$\sum_{k=1}^N \frac{X_k^2}{N} < \text{threshold} \quad (\text{しきい値})$$

ここで、2乗の和が所定のしきい値より小である場合には、テキスト要素が接している線要素と判別する。この際、しきい値として、テキスト要素が接していない線の検出におけるしきい値を用いると、良好な結果が得られる。ステップS435において線を検出すると、処理をステップS436からステップS449に進め、「線」属性を非テキスト要素に付与する。そして、処理をステップS407に戻し、次の要素を選択する。

【0080】一方、ステップS435で線が検出されなかった場合には、処理をステップS436からステップS437に進め、非テキスト要素のサイズを調べる。ここで、サイズが所定のしきい値以下であると、非テキスト要素の分類が不可能となる。なお、このしきい値は最大のフォントサイズに基づいて決められるものであり、最大フォントサイズの半分の値で良好な結果が得られる。そして、処理をステップS438に進めて「未知」属性を非テキスト要素に付与し、処理をステップS407に戻して次の要素を選択する。

【0081】ステップS437においてサイズが所定の



しきい値よりも大であった場合には、処理をステップS 439、S 440、S 441に進め、ステップS 417、S 418、S 419で述べたように、非テキスト要素内部の白輪郭を追跡し、非テキスト要素の密度を求め、白輪郭の数を調べる。ステップS 441において白輪郭の数が4以下であれば処理をステップS 442に進め、要素サイズを計算し、要素が線画や中間調画像であるのに十分な大きさであるかを調べる。このサイズ計算は、非テキスト要素の高さと幅ならびに黒画素の最大ラン長に基づいて行われる。具体的には、非テキスト要素の高さや幅が最大フォントサイズ以下であれば、非テキスト要素は中間調画像や線画であるほど大きくはないと判断し、処理をステップS 443に進め「未知」属性を付与する。また、非テキスト要素の幅は最大フォントサイズより大であるが、黒画素の最大ラン長は最大フォントサイズ以下である場合にも、処理をステップS 443に進め、「未知」属性を付与する。そのあと、ステップS 407に処理を戻し、新たな要素を選択する。

【0082】ステップS 442において非テキスト要素が線画や中間調画像であるのに十分な大きさであると判断されると、ステップS 444に処理を進め、非テキスト要素が線画であるか中間調画像であるかの判別を行う。ステップS 444からステップS 448までの処理は、それぞれステップS 428からステップS 432までの処理と同一のものであり、ここでの説明は省略する。

【0083】図4A～図4C（図3中のステップS 301）を参照して以上説明したように、画素画像中のすべての連結要素を検出して分類すると、図14に示されるような木構造が得られる。図14に示されるように、木構造の根は画素画像データのページに対応する。根からの子ノードには、テキストブロックや、未知、フレーム、絵、線などからなる非テキストブロックが存在する。また、フレームの子ノードには、テキストブロック、未知の非テキストブロック、テキストブロックを有する表、絵、線などが存在する。

【0084】図10は画素画像データの典型的なページ90を示したものであり、大きなフォントサイズのテキスト91、テキスト93などを含む表92、テキストデータ94、水平線95、もう一つのタイトル96、2段落のテキストデータ97及び見だし99を備えるフレーム線画98とタイトル100で始まり、テキストデータ101、見だし103を備えるフレーム中間調画像102、テキストデータ104、水平線105、テキストデータの最終段落106へと続く第2段とが示されている。図11は、ステップS 301に基づく処理後の同一画素画像を示したものである。図11に示されているように、画素画像データ90中の連結要素が矩形ブロックで囲まれており、矩形ブロックの内部についてはステップS 415からステップS 434までのフレーム処理で

判別される。

【0085】ステップS 302では、ステップS 301で得られたすべてのテキスト要素を、木構造の位置に関わらず水平方向にグループ化する。グループ処理は各テキスト要素の密集性ならびにその近傍関係に基づいて行われる。なお、この際、段に対応する垂直方向に伸びたギャップが検出され保持される。以下、図8を参照して、ステップS 302で行う詳細な処理について説明を加える。図8に示される処理ステップは、ROM 17に保持されるプログラムステップに基づいてCPU 10で実行される。

【0086】ステップS 801では、非テキスト要素の左端ならびに右端から垂直にギャップラインマーカを伸ばす処理を行う。すなわち、図11に示されるように、ギャップラインマーカ109aと109bとをテキストあるいは非テキスト要素（ここでは要素95）に達するまで垂直に伸ばす。また、ギャップラインマーカ109cと109dとをテキストあるいは非テキスト要素（ここでは要素95）に達するまで垂直方向に伸ばす。同様に、残りの非テキスト要素の左端ならびに右端からギャップラインマーカを垂直に伸ばす。このようなギャップラインマーカを用いることで、画素画像データにおいて段に対応するギャップ位置の判別が容易になる。

【0087】ステップS 802において、図11のテキスト要素107のようなテキスト要素は、連結によりギャップラインマーカを横切らず、且つ他のテキスト要素と接しているかあるいは他のテキスト要素から所定のしきい値内の距離にあれば、1つのテキストラインに連結される。なお、適切なしきい値として、ステップS 404で求めた平均テキスト長に経験的に得られたスカラー値を掛けたものを用いる（「1.2」を用いると良好な結果が得られる）。また、連結処理を行うに先立って、テキスト要素間の垂直ギャップを調べ、段構造を示唆するような垂直ギャップが存在するかどうかを判別する。すなわち、図11に示されるように、ギャップ108はテキスト要素のペアの間に存在するが、このギャップはテキスト画像データ中で垂直方向に数ライン程度の長さとなるため、テキスト要素が他の要素から所定のしきい値内の距離に位置しているにも関わらず、ステップS 802ではギャップが維持される。

【0088】ステップS 803では、連結ステップS 802で連結されなかったテキスト要素のペアが連結によりギャップラインマーカを横切らず、隣のラインの第3テキスト要素と共に重なるような場合に、これらのテキスト要素のペアを連結する。このようなステップにより、段構造を示すギャップではなく単にテキストライン中のランダムな空間構成に起因するようなギャップを効果的に除去することが可能となる。例えば、図11において、ステップS 802ではギャップ108の連結は行われないが、ギャップ両側のテキスト要素は1行下のラ

インの第3テキスト要素と重なり、またギャップラインマーカを横切らないため、ステップS803においてこのギャップが除去される。

【0089】ステップS804では、これらの処理結果に基づいて木構造の適切な更新が行われる。図12は、ステップS302のグループ化処理の結果を示しており、図15はステップS302のグループ化処理で修正された木構造を示している。図12に示されるように、各々が接しているテキスト要素は、ライン110のようにテキストラインにグループ化される。すなわち、木構造中に位置するテキスト要素をテキストラインに連結する処理を行うが、111などのようにテキスト要素が木構造中のフレーム表ノードの下に位置する場合にも連結処理が行われる。ここで、このようなグループ化処理は、上のステップS417からS439で求めた白輪郭境界を横切ることはなく、表中の個々の項目が1つの連続するテキストラインにグループ化されることはない。また、左段と右段との間のギャップは維持される。さらに、非テキスト要素は再連結されない。すなわち、112や113などの要素のように、互いに所定のしきい値内の距離にあっても、非テキスト要素のグループ化は行わない。

【0090】図15には、新たなグループ化処理の結果に基づいて修正した木構造が示されている。図8（図3中のステップS302）を参照して説明したようにテキスト要素のテキストラインのグループ化を行った後には、ステップS303に示されるようにテキストラインを垂直方向にグループ化してテキストブロックを生成する処理を行う。

【0091】以下、図9を参照してこの処理について詳述する。グループ化処理は、テキストライン要素の密集性や非テキスト要素の位置に基づいて行われる。例えば、間に存在する非テキストラインは境界を示すため、これを利用すれば非テキストライン両側のテキストラインを1つのテキストブロックにグループ化することを避けられる。なお、処理は、2つの連続する非テキストライン要素間のすべてのテキストラインに対して1度に行われる。また、ステップS303では、テキスト要素を非テキスト要素に連結すべきか（例えば、非テキスト画像とそのテキスト見だし）、非テキスト要素を他の非テキスト要素に連結すべきか（例えば、中間調画像と線画）の判断も行う。

【0092】図9は、テキストラインのテキストブロックへのグループ化を示す詳細なフローチャートである。ステップS901においては、最大の予想フォントサイズより小ではあったが平均テキストサイズより大であったため、ステップS404で非テキスト要素に分類された非テキスト要素からタイトルブロックの形成を行う。隣り合う非テキスト要素のうち同じようなサイズの要素に対してはすべてグループ化を行い、タイトルブロック

を形成し、「タイトル」属性をこのグループに付与する。ここでグループ化されなかった残りすべての非テキスト要素に対しては「絵-テキスト」属性が付与される。また、それに応じて木構造の更新も行う。ここで得られたタイトルは、ページ再生時（ステップS212）に有用な情報となる。

【0093】ステップS902では、2つのテキストラインにまたがる非テキスト要素の位置を明確にする。このような非テキスト要素はテキストブロック間の境界となり、テキストラインを1つのテキストブロックにグループ化してしまうことを避けることができる。ステップS903では、2つのステップでテキストラインの垂直方向のグループ化を行い、テキストブロックを形成する。第1のステップでは、画素密度の垂直ヒストグラムを求めるなどして、段間のギャップを検出する。第2のステップでは、垂直方向に連続するテキストライン間の垂直距離がステップS404で求めたテキストの高さより小さければ、それぞれの段ごとにテキストラインのグループ化を行う。このステップS903の処理により、図2のライン114のような同一テキスト段落中のテキストラインを、テキストブロックにグループ化する。

【0094】ステップS904では、垂直ならびに水平方向に隣接するテキストブロックのグループ化処理を行う。ここで、これらのテキストブロックが非テキスト要素で分離されていず、またこれらのブロックを連結してもステップS903のヒストグラムから求めたギャップが保持されるときに、テキストブロックのグループ化が行われる。また、ブロック間の距離がステップS404の垂直の高さに基づいて計算される所定のしきい値より小であるときに、テキストブロックはグループ化される。このステップS904の処理により、図12中の段落115のラインと段落116のラインのテキストブロックはグループ化されるが、段落117と118のラインのテキストブロックはそれらの間に非テキスト要素119（線）を有するためグループ化が行われない。

【0095】ステップS905では、テキストブロックを非テキストブロックに連結すべきか、非テキストブロックを他の非テキストブロックに連結すべきかを判断する。ここで、テキストブロックの非テキスト-タイトルブロック、非テキスト-中間調ブロック、非テキスト-ライン接触ブロックとの連結は、以下に行われる。

(1) テキストブロックが非テキスト-タイトルブロックと水平方向に近い位置にあり、垂直方向に重なっている場合には、テキストブロックを非テキスト-タイトルブロックに連結する。

(2) テキストブロックがワードサイズのブロックより小さくて（水平、垂直方向とも）、テキストブロックの隣にワードサイズのテキストブロックが存在しないときには、テキストブロックを非テキスト-中間調画像プロ



ックの中に位置付ける。

(3) テキストブロックが非テキストライン接触ブロックと重なっていれば、ライン接触ブロックはテキストの下線である可能性が高いため、ライン接触ブロックをテキストブロックに変換する。

\*

\*【0096】また、以下の表にしたがって、非テキストブロックは他の非テキストブロックと連結される。

【0097】

【表1】

|        | 中間調  | 線画   | テキスト-絵 | タイトル |
|--------|------|------|--------|------|
| 中間調    | テスト1 | 非連結  | 常に連結   | 非連結  |
| テキスト-絵 | テスト1 | テスト2 | テスト2   | テスト3 |
| 線画     | テスト1 | テスト1 | 非連結    | 非連結  |
| タイトル   | 非連結  | 非連結  | 非連結    | テスト3 |

この表中のテストは以下の通りである。

テスト1：1つのブロックが完全に他のブロック内に位置すれば連結。

テスト2：絵-テキストの幅がワードサイズのブロックの幅より小さければ連結。

テスト3：ブロックが近接していれば連結。

【0098】ステップS906では、適切な属性が付与され、上述の処理結果に基づいて木構造の更新が行われる。図13は図9の処理を行った結果のブロック構造であり、図16はその木構造の例である。図13において、ブロックには、タイトルブロック120、テキストブロック121及び絵データ122が含まれる。また、フォームデータも含まれ、123は表構成のデータを有するフレーム要素を示しており、124はテキスト要素125を有するフレーム要素を示している。なお、非テキストライン画像127は図13に示されるさまざまな要素を分離している。

【0099】図3から図16を参照しながら説明したブロック選択処理を終えると、上述のように文字認識処理は図2のステップS204に処理を進める。すなわち、階層の木構造中の第1ブロックを選択して認識処理を行う。このブロックがテキストブロックでない場合には、処理をステップS204からステップS205に進め、木構造中の次のブロックを選択する。テキストブロックが選択されるまでステップS204とS205を繰り返し、テキストブロックが選択された時点でステップS2※

※06に処理を進め、ラインの分割処理を行う。

【0100】図17は、図2のライン分割ステップS206で実行される処理ステップを詳細に示すフローチャートである。図17に示される処理ステップは、プログラムROM17に保持されるコンピュータプログラムにしたがってCPU10で実行される。ステップS1701に先立って画像縮小処理を行っても良い。しかし、ライン分割処理や文字分割処理は水平方向の空白により影響を受けやすいため、画像縮小処理を行うにあたっては分割精度に影響を与えないような注意が必要である。すなわち、水平方向と垂直方向とでそれぞれ異なる画像縮小手法を用いることが好ましい。垂直方向では画素の結合を「OR」論理で行い、垂直方向の対象画素のうち1つでも黒画素が存在すれば黒画素が出力される。すなわち、垂直方向の2：1の画像縮小処理では、2つの垂直画素のいずれかが黒画素であれば黒画素が出力される。これに対して水平方向では画素の結合を「AND」論理で行い、水平方向の対象画素すべてが黒画素であれば黒画素が出力される。すなわち、水平方向の3：1の画像縮小処理では、3つの画素がすべて黒画素のときのみ黒画素が出力される。

【0101】垂直方向に3：1の縮小を行い、水平方向に2：1の縮小を行う場合の処理例を以下に示す（「0」は白画素を「X」は黒画素を表す）。

【0102】

【表2】

| 原画像  | 垂直縮小(OR)           | 水平縮小(AND)  |
|--|--------------------|------------|
| X O X O<br>X X X O<br>O O O O<br>O X O O<br>O X X X<br>O O X O | X X X O<br>O X X X | X O<br>O X |

画像縮小処理を終えると、この縮小画像に対してライン分割処理と文字切り出し処理とが行われる。ここで、図2の残りの認識処理（すなわち、ステップS208からステップS213まで）に影響を及ぼさないように、ラ

イン分割処理と文字切り出し処理とを終えた時点で、文字間の切り出しは縮小していないもとの画像データに対して行う。

【0103】ステップS1701では、ステップS20

4で選択されたテキストデータブロックごとに画素密度の水平投影を求める。画素密度の水平投影は、画素画像の各行ごとに黒画素数を計数して得られる。ここで、画素密度の水平投影は全体のテキストブロックに対して求めることが好ましいが、これは本質的な点ではない。すなわち、テキストブロックを複数の画素列、例えば2あるいは3列に分割して、各列ごとに画素密度の水平投影を求めることができる。もちろん、このような処理では、本発明の処理時間短縮という利点は失われる。

【0104】ステップS1702では、水平投影のうち10  
ゼロでない領域を求め、どれかが最大フォントサイズと等しい所定のしきい値より大であるか否かを調べる。最大フォントサイズ以下であれば、水平投影はページ上のラインを均一に分割していることになり、処理をステップS1703に進める。ステップS1703では、画素密度の水平投影のうち近接している領域を連結する。この処理を図18A～図18Dを用いて説明する。図18Aは典型的なテキストブロック230を示しており、文字画像のライン231と233、ならびに雑音スポット232（すなわち、文字情報ではない黒画素）が含まれる。これに対応する画素密度の水平投影は234に示される。水平投影234に示されるように、領域235は文字「I」の上の点に対応し、領域236はライン231上の残りの文字に対応し、領域237と238は雑音スポット232に対応し、領域239はライン233上の文字に対応する。これらの各ピークにより、境界が水平投影のゼロ値となる領域が定義される。ステップS1703では、同一テキストライン上に存在するピーク235と236のような近接領域は連結され、テキストラインとは関係のないピーク237と238のような近接領域は連結されないことが望まれる。20

【0105】ステップS1703で述べたように、近接領域を連結するためには、画素密度の水平投影を投影の上部から下部まで（テキストブロックの上部から下部まで）調べる。第1の領域を検出すると、その下の次の領域の水平投影をスキャンして、2つの領域の高さを比較する。上の領域の最も高い要素の高さが下の領域の高さより小さく、2つの領域の間隔が上の領域中の最も高い要素の高さより小さければ、これら2つの領域を連結する。ここで、この近接領域を連結する処理は、スケール20  
に対して不変であることに注意されたい。すなわち、テキストの近接領域を連結する際には、テキストサイズが12ポイントであるとか8ポイントであるなどの知識は不要である。

【0106】そこで、図18Aに戻り、領域235の高さと領域236の高さとの比較を行うと、上の領域235の高さが下の領域の高さより小さいことがわかる。また、2つの領域間のギャップは領域235の高さより小さいと判断される。したがって、図18Bに示されるように、領域235と236とは1つの領域236'とし20

て連結される。

【0107】画素密度の水平投影を下方向に調べると、領域237が検出される。この場合、領域236'の高さが領域237の高さ以上であるため、これら2つの領域は連結されない。さらに、画素密度の水平投影を下方向に調べると、領域237の高さが領域238の高さ以下であり、また領域間のギャップは領域237の高さ以下であることがわかる。そこで、図18Cに示されるように、領域237と238とは1つの領域238'として連結される。画素密度の水平投影をさらに下方向に調べると、領域239が検出される。この場合、領域238'の高さは領域239の高さより低いが、領域間のギャップは領域238'を構成している領域237と238とのどちらかの高さより大きいことがわかる。そこで、これら2つの領域は連結されない。

【0108】また、近接領域の連結処理のあとに各領域の高さを調べて各領域の高さがラインの最小の高さに対応するしきい値より大であるかどうかを判断することもできる。この際、しきい値は、これまでに検出された領域の高さの平均として適当に設定される。領域がしきい値以下の高さの場合には、領域を画素データ中の雑音スポットに起因するものと判断し削除することができる。したがって、領域238'の高さは、領域236'、238'、239の高さの平均として決められるしきい値以下であるため、図18Dに示されるように領域238'は削除される。

【0109】これらの処理を終えると、図17に戻って処理をステップS1704に進め、領域を個々のラインセグメントに分割し、図2のステップ207で示される文字切り出しに処理を進める。ステップS1702において、ステップS1701で処理された領域が大きすぎる領域であった場合には、テキストラインが傾いていると判断される。例えば、図19Aに示されているように、テキストブロック240には複数の傾いたテキストライン241が含まれる。ステップS1702に基づく処理では、244に代表テキストを示したようにテキストラインが水平方向に相互に重なりあってしまうため、242のような画素密度の水平投影が得られる。

【0110】そこで、処理をステップS1705に進め、テキストブロックを段に分割する。図19Bに示されるように、テキストブロックの段数を2倍にする。すなわち、テキストブロック240を2つの段に分割する。この際、少なくとも1つの共通の画素が重なり合うように段を分割することが好ましい。また、テキストブロックを2ブロック以上、例えば3あるいは4ブロックに分割することも可能である。

【0111】ステップS1706では図19Bの247や249のように各段ごとに画素密度の水平投影を求め、ステップS1707で領域が大きすぎるかどうかを再び調べる。領域が大きすぎる場合には、処理をステッ

ブS1708に進め、段数を再び増加させる。例えば、図19Cに示されているように、段数を更に2倍にする。また、ステップS1709において、段の幅が最低限度より大であるかどうかを確認する。この最低限度は、これ以上段数を増加させると適切なライン分割が不可能になる点を示している。好適な実施例では、最低限度は16画素幅である。ステップS1709において最低限度に達していると判断されると、ライン分割は不可能であると表示して処理を終了する。一方、最低限度まで達していない場合には、処理をステップS1706に戻し、新たな段ごとに水平投影を再び計算する。ステップS1707において領域が大きすぎないと判断された場合には、処理をステップS1710に進める。すなわち、図19Cに示されるように、ラインセグメント以下の幅の領域が割り出された。そこで、処理をステップS1710に進め、ステップS1703で説明したように近接領域を連結する。そして、ステップS1711において、各段ごとに単一ラインセグメントに対応する領域を割り出す。

【0112】すなわち、図19Cにおいて、単一ラインセグメントに対応する領域250、251、252、253を検出する。そして、領域が異なる段間で接触しており単一ラインセグメントを構成するかどうかを判別するために、各段を上から下に調べ各段の第1領域を検出する。そこで、その領域に接触している領域を調べ、図19Dに示されているように2つの距離を求める。この2つの距離、(1)2つの領域を合わせたときの全長距離Aと、(2)2つの領域の共有領域すなわち2つの領域の交差領域の距離Bとの2つである。そして、比A/Bを求め、2つの領域が多くの部分で重なっているかを確かめるためにしきい値と比較する(しきい値として5を用いると良好な結果が得られる)。比A/Bがしきい値より小であれば、2つのブロックは多くの部分で重複していることになり、ブロックは単一ラインセグメントを構成していると考えられる。そこで、ステップS1712において、比A/Bで求められる接触領域を単一ラインセグメントとして割り出される。

【0113】ここで、比A/Bの計算と、比A/Bとしきい値との比較処理とは、スケールに不変な処理であり、ライン中のテキストサイズに関わらず、重なった接触領域は単一ラインセグメントとして割り出される。このようなスケール不変の性質は、ラインセグメント中のテキストサイズが既知である必要がないという点で望ましいものである。

【0114】図17で述べたラインの分割処理を終えると、図2のステップS207で説明し、詳細が図20に示される文字分割(文字切り出し)に処理を進める。図20に示されるように、文字切り出しは多階層の処理で行われ、各階層は徐々に複雑な文字切り出し処理を行う。すなわち、文字切り出し処理は3つの処理に分類さ

れる。相互に接触ならびに重なっていない文字間の切り出し、相互に接触してはいないが重なっている文字間の切り出し、接触している文字間の切り出しの3つである。例えば、図34Bに示されるように、文字「S」と「a」は接触ならびに重なっていないため第1の処理に分類される。一方、文字「f」と「y」は接触してはいないが重なっているため第2の処理に分類される。さらに、文字「t」と「l」は相互に接触しているため第3の処理に分類される。

【0115】図20に示されるように、各階層は3つの処理のうちの1つを実行するように構成される。すなわち、階層1(261)では接触ならびに重なっていない文字間の切り出しが行われる。階層1に続いて、当該テキストの性質及び特徴に関する知識の有無に応じて処理を進める。テキストが単一スペースのテキストであれば、すなわち文字が垂直で文字間が均一であれば(例えば「クーリエ」フォント)、処理を階層2(262)に進め、接触している文字の切り出しを行う。単一スペースの文字であっても、コピーやファクシミリ転送の繰り返しに起因する画像の劣化のため、文字が相互に接触することがある。そして、文字認識処理263に処理を進め、図2のステップS209に進む。

【0116】一方、テキストブロックの性質及び特徴に関して何の情報もない場合、あるいはテキストが単一スペースのテキストでなければ、処理を階層2(264)に進め、接触はしていないが重なっている文字間の切り出しを行う。そして、階層1と階層2との切り出しで得られた文字に対して265で認識処理を行う。認識処理265で認識されなかったすべての文字に対しては、階層3(266)の処理を行う。すなわち、文字が認識されない理由として、文字の切り出しが完全に行われないため実際には認識不能な文字は接触している複数の文字と考えられる。そこで、階層3では接触している文字の切り出しが行われる。階層3で文字を切り出すと、267で文字認識処理を行う。認識処理が成功した場合には、処理を図2のステップS209に戻す。一方、認識処理が再び失敗した場合には、階層3での切り出しが不適切であったと判断する。そこで、認識不能であった切り出しを再接続し、再び階層3の切り出し処理と認識処理とを行う。このような処理を、文字の切り出しが不可能になるまで繰り返し行う。

【0117】「単一スペース(mono-spaced)」262か「全スペース(all-spacing)」264かのどちらに処理を進めるかは、オペレータからの入力によって選択される。オペレータからの入力がない場合には、デフォルトとして処理を「全スペース」264に進める。というのは、この階層は単一スペースのみならず非単一スペース文字にも適用できるためである。

【0118】図21から28までは階層1から階層3までの処理を説明する図であり、図30と図31は269

で示される再接続処理を説明する図である。図21は非接触で重なっていない文字を切り出す階層1の切り出し処理を説明するフローチャートである。階層1では、2つの文字間の白画素すなわち空白を検出することにより、非接触で重なっていない文字の切り出しを行う。

【0119】具体的には、図21のステップS2101に示されるように、空白でない画素すなわち黒画素が検出されるまでラインセグメント中をとびとびに検索し、文字間の白スペースを検出する。ここで、とびとびの検索とは、ラインセグメントのすべての画素を検索するのではなく、図22に示されるように、ラインセグメントの1つの段中の一部の画素271のみを検索するものである。なお、ラインセグメントの段中の全画素の1/3のみ、すなわち3画素ごとの検索で十分であるとの結果が得られている。画素271中で空白でない画素すなわち黒画素が検出されなかった場合には、272で示すように数列の画素をとびとびとして、例えば3画素ごとに新たな列で黒画素を検索する。ここで、画素を3列とびとびとして、非接触で重なっていない文字の検出性能は低下しないとの結果が得られている。このようなとびとびの検索を、図22の画素274のような最初の黒画素が検出されるまで繰り返す。

【0120】最初の黒画素が検出されるとステップS2102に処理を進め、ラインセグメントを後向きに検索し、完全に空白な列を検出する。ここでの検索は、ステップS2101の検索とは異なり、各列ごとにすべての画素を検索して完全に空白な列を検出するものである。すなわち、図22に示されるように、完全に空白な列276が検出されるまで後向きステップ275が実行される。

【0121】完全に空白な列が検出されると、処理をステップS2103に進め、完全に空白な列が検出されるまで画素274の列から前向きに検索される。ステップS2102と同様に、ここでの前向き検索も各列ごとにすべての画素を検索するもので、277で示すように前向きに検索される。図22の278で示されるような完全に空白な列が検出されるまで前向き検索が行われる。

【0122】ステップS2103で完全に空白な列が検出されると、処理をステップS2104に進め、空白列276と278とで文字を切り出す。その後、処理をステップS2101に戻し、空白でないすなわち黒画素が再び検出されるまで、ラインセグメントでのとびとびの検索を再び行う。ラインセグメント全体で階層1の処理を終えると、テキストが単一スペース（クーリエフォントなど）であるか、あるいはテキストのスペースが未知もしくは単一スペース以外（比例フォントなど）であるかに応じて、図20の261あるいは264の階層2の処理に進める。テキストが単一スペースであれば、261の階層2の切り出しに処理を進める。

【0123】図23は単一スペーステキストのための階

層2の処理の流れを示すフローチャートである。なお、図23に示される処理ステップは、ROM17に保持されるプログラムステップに基づいてCPU10で実行される。階層2の処理に先立って、文字セグメントの幅を調べて、過小サイズの文字セグメントを割り出す。ここで、文字セグメントの幅が平均文字幅の半分以下のときに過小サイズであると判断する。過小サイズの文字セグメントが隣り合って存在する場合には、階層1の処理で1つの文字を半分ずつ2つに切り出してしまった可能性が高いため、この過小サイズ文字のペアを連結する。

【0124】ステップS2301では、各文字ブロックの幅をすべての文字ブロックの平均幅と比較して、階層1で切り出された文字ブロックのうち過大サイズのものを割り出す。ここで、各文字ブロックの幅とすべての文字ブロックの平均幅との比較は、文字が単一スペースであって各文字がほぼ同一の幅を有することが既知であるため、過大サイズの文字ブロックの検出には有効な処理となる。文字ブロックの幅（「W」）が以下の式を満たすと、文字ブロックが過大サイズであると判断される。

【0125】

$$[数5] W > (1+c) * W_{ave}$$

ここで、cは定数、 $W_{ave}$ はラインセグメント中のすべての文字ブロックの平均幅である。なお、この判別処理はスケールに不変であることに注意されたい。定数cは、以下のように単一スペースフォントの統計的性質に基づいて決定される。クーリエアルファベットなどの単一スペースアルファベットの各文字は、単一スペースを有しており、各スペースは文字が存在する部分 $\alpha_i$ と文字の周りの空白スペースの部分 $\beta_i$ とからなる。

【0126】例えば、図24の文字「e」に示されるように、文字「e」が存在するスペースは中心領域 $\alpha_i$ と周りの空白スペース $\beta_i$ とからなる。ここで、iは「e」に対応する番号であり、すべてのiに対して $\alpha_i + \beta_i = 1$ である。 $\alpha_i$ と $\beta_i$ についてはアルファベット中の各文字、すなわち英字、数字、記号などごとに求めることができ、 $\alpha_i$ と $\beta_i$ の平均値（それぞれ $\alpha$ と $\beta$ ）ならびに標準偏差（それぞれ $\sigma_\alpha$ と $\sigma_\beta$ ）を計算できる。そこで、定数cは以下の式で求める。

【0127】

$$[数6] c = \sigma_\alpha / \alpha$$

図1の装置で用いるクーリエ文字セットでは、 $\alpha = 25/35$ 、 $\sigma_\alpha = 10/35$ となるため $c = 0.4$ となる。ステップS2301で過大サイズの文字ブロックが割り出されると、ステップS2302に処理を進め、過大サイズブロック中に含まれるおおよその文字数を算出し、おおよその文字境界を求める。具体的には、図24において、ブロック280の幅Wは、すべてのブロック280から283の平均の幅に $(1+c)$ を乗算して計算されるしきい値より大であるため、ブロック280は過大サイズの文字ブロックと判断される。そして、幅W

を $\alpha$ で割った値を最も近い整数値に丸めて、過大サイズブロック280中のおおよその文字数を算出する。

【0128】

【数7】文字数 $N = [W/\alpha]$  (最も近い整数)

また、ここで得られたブロック中のおおよその文字数「N」に基づいて、過大サイズのブロックを単一に分割して、おおよその文字境界を求める。ステップS2303では、ブロック中の画素の垂直投影特性284を、おおよその文字境界の近傍285で求める。ここで、垂直投影特性284を求める近傍は、距離 $\sigma$ に基づいて決

められる。すなわち、図24に示されるように、おおよその文字境界の近傍 $\pm\sigma$ で垂直投影特性284を求める。【0129】ステップS2304では、各垂直投影特性284中での最小位置286を割り出し、この最小位置284で文字の切り出しを行う。図20の261の階層2の処理を終えると、文字認識263、そしてさらに図2のステップS209に処理を進める。ラインセグメント中の文字のスペースが未知あるいは単一スペースでない場合には、文字がラインセグメント中で単一のスペースを有しているとは限らない。そこで、図20の264の階層2の処理に進み、非接触であるが重なっている文字間の切り出しを行う。図25と図26とはこの処理を説明する図である。

【0130】ステップS2501において、階層1で切り出された各文字を分析して、文字ブロック中の各画像の輪郭のアウトラインを追跡する処理を行う。すなわち、図26Aに示されるように、文字「fy」を有する文字ブロックは非接触だが重なっている文字「f」と「y」を含み、これらは重なっているため階層1の処理では切り出されない。そこで、図26Bに示されるように、まず、この文字ブロックをブロックの右下から左方向ならびに上方向に調べて黒画素を検出する。黒画素を検出すると、図26Cの287のように黒画素と黒画素とを接続したものである輪郭の追跡が行われる。第1の文字に対して全体の輪郭追跡を終えると、288のように文字ブロック中のすべての黒画素の輪郭を追跡するまで、スキャン処理を続ける。このような処理により、各々が分離した非接触の文字が得られ、図26Dに示されるようにこれらの文字が文字ブロックから切り出される。

【0131】階層2の処理では、非接触だが重なっている文字の切り出しとともに、複数のストロークからなる単一文字、例えば「i」、「j」、「:」、「;」、「!」、「=」、「%」をも分離してしまう。そこで、ステップS2502でこのような文字の再接続を行う。図27はこの処理の詳細なフローチャートである。図27に示す再接続処理の対象となるのは階層2の処理で切り出された文字のみであり、特定の条件を満たすときのみ再接続が行われる。具体的には、ブロックが重なる、

すなわち左側文字の最も右側の画素が右側文字の最も左側の画素の上あるいは下に位置するような場合にのみ、ブロックの再接続が行われる。

【0132】そこで、ステップS2701でブロックが重なっているかどうかを調べる。ブロックが重複していなければ、再接続処理は不必要であり(ステップS2702)、再接続処理を終了する。一方、ブロックが重複していれば、ステップS2703に処理を進め、ブロックが垂直方向に分離しているかどうかを調べる。ここで、ブロックが垂直方向に分離していると、「i」、「j」、「:」、「;」、「!」、「=」などの複数ストローク文字が階層2の処理で切り出されてしまった可能性があるため、これらの文字であるかどうかを調べる。

【0133】これらのブロックは垂直方向に分離されているため、高さがH1の上部ブロックと高さがH2の下部ブロックとを含む。これらの高さの計算はステップS2704で行われ、ステップS2705でH2が(2×H1)より大であれば文字「i」あるいは「j」が分離したものである可能性が高い。そこで、文字の重複度を計算する(ステップS2706)。すなわち、2つの部分に隣接する最も右側の4画素列を平均して、この平均位置の差を計算する。(全体位置ではなく最も右側の位置を平均したのは、文字「i」や「j」の上のドットが「i」や「j」の中心ではなく上部のセリフの右側に位置するためである)。この際、スケール不変の処理とするために、平均位置の差が本体のうちの小さい幅に定数を乗じたものより小であれば、ブロックを再接続するものとする。ここで、定数は、分離の程度が予測できないような劣化画像でも再接続できるように選択され、本実施例では「9/8」としている。

【0134】一方、ステップS2705でH2が(2×H1)以下、すなわち下部ブロックの高さが上部本体の2倍以下である場合には、「:」、「;」、「!」、「=」などの文字が切り出された可能性が高い。そこで、ステップS2707に処理を進め、この可能性を調べる。すなわち、各本体中の隣接する4画素列の平均中心値を求め、これらの中心値の差を得る。この際、スケール不変の処理とするために、中心値の差が2つの本体の幅のうちの小さい方に定数を乗じたものより小であれば、上述の文字の1つである可能性が高いためブロックを再接続するものとする。なお、上述のように、定数として「9/8」を用いると良好な結果が得られている。

【0135】ステップS2703でブロックが垂直方向に分離されない場合(すなわち、2つのブロック間に水平方向に伸びるギャップが存在しない場合)には、文字はタイプ1のパーセント記号(「タイプ1」)、タイプ2のパーセント記号(「タイプ2」)、タイプ3のパーセント記号(「タイプ3」)のどれかである可能性が高い。そこで、ステップS2708で以下のように順々に

各タイプごとにチェックする。なお、変数は以下の通りである。

【0136】W1：第1文字の幅（左から右）

W2：第2文字の幅

H1：第1文字の高さ（上から下）

H2：第2文字の高さ

L1：第1文字の左端画素の列

R1：第1文字の右端画素の列+1画素

L2：第2文字の左端画素の列

R2：第2文字の右端画素の列+1画素

注意：L1は常にL2以下である。

【0137】まず、タイプ1のパーセント記号をチェックする。以下の2つの条件が満たされるときにタイプ1のパーセント記号であると判断され、ブロックが連結される。

1)  $0.24 < \min(W1, W2) / \max(W1, W2) < 0.77$

これはドット幅とライン幅との比の条件である。

【0138】2)  $[\min(R1, R2) - \max(L1, L2)] / \min(W1, W2) > 0.76$

これはブロックが水平方向に大部分重なっているという条件である。次いで、タイプ2のパーセント記号をチェックする。以下の4つの条件が満たされるときにタイプ2のパーセント記号であると判断され、ブロックが連結される。

【0139】1)  $(0.25) L2 < R1 - L2$

これはブロックが水平方向に十分重なっているという条件である。

2)  $0.50 < W1/W2 < 1.10$

これはドット幅とライン幅の適切な比率の条件である。

3)  $0.43 < (H1/H2) < 0.70$

これはドット高とライン高の適切な比率の条件である。

【0140】4)  $(1/m) > 0.37$

ここで、mはパーセント記号の「斜線」部上のP1とP2とを結ぶ線の傾きである。なお、P1とP2とは以下の手法で求められる。

P1：P1は第2文字の上部からD行目の行で、第2文字のプリントテキストを含む左端の画素の位置である。ここで、変数DはD=(0.1)W2である。

【0141】P2：P2は第2文字の下部からD行目の行で、第2文字のプリントテキストを含む左端の画素の位置である。

さらに、タイプ3のパーセント記号をチェックする。以下の条件が満たされるときにタイプ3のパーセント記号であると判断され、ブロックが連結される。

1)  $(0.25) L1 < R2 - L1$

これはブロックが水平方向に十分重なっているという条件である。

【0142】2)  $0.50 < W2/W1 < 1.10$

これはドット幅とライン幅の適切な比率の条件である。

3)  $0.43 < (H2/H1) < 0.70$

これはドット高とライン高の適切な比率の条件である。

4)  $(1/m) > 0.37$

ここで、mはパーセント記号の「斜線」部位上のP1とP2とを結ぶ線の傾きである。なお、P1とP2とは以下の手法で求められる。

【0143】P1：P1は第1文字の上部からD行目の行で、第2文字のプリントテキストを含む右端の画素の位置である。ここで、変数DはD=(0.1)W2である。

P2：P2は第1文字の下部からD行目の行で、第2文字のプリントテキストを含む右端の画素の位置である。

図20の264で説明した（また、図23から図27で詳細に説明した）階層2の切り出し処理と再接続処理を終えると、265の認識処理を切り出された文字に対して行う。階層1と階層2の切り出しにおいてラインセグメント中のほとんどの文字は適切に切り出されているため、265の認識処理では階層1と階層2で切り出された文字のほとんどを認識することができる。これに対し、265で認識不能であった文字は、その文字ブロックに接触文字が含まれている可能性が高い。そこで、このような認識不能の文字ブロックに対して、266の階層3の切り出し処理を行い、接触文字を切り出す。

【0144】図28は階層3の切り出し処理を示すフローチャートであり、図29Aから図29Dは接触文字を階層3で切り出す処理を説明する図である。図28に示される処理ステップはROM17に保持されており、CPU10で実行される。一般に階層3の切り出し処理は、文字ブロックを斜めに切り出して行われる。斜めの切り出し線の傾きと位置とは、ブロック中の画素密度の垂直投影特性を求め、垂直投影特性中で最も深い谷の側面の傾きを求めることで算出される。そこで、再び画素密度の垂直方向以外の投影を行う。すなわち、垂直投影特性中の谷側面の傾きに対応する回転角度方向に画素密度の投影を行う。こうして得られた複数の密度投影の中での最小点を検出し、最小点が得られた角度と位置で切り出しを行う。以下、この処理について詳細な説明を行う。

【0145】ステップS2801において、画素密度の垂直投影特性を求める。例えば、図29Aに示されるように、接触文字「t1」に対して垂直投影特性を求める。ステップS2802では、垂直投影特性中の第1の谷を検出する。垂直投影特性はデジタルであるため（すなわち、離散的な画素数の和であるため）、滑らかではなく、谷は垂直投影特性中の最小値が第1の低い値より下であって、第2の高い値より大きい極大点で両側が囲まれた点であることで見付け出される。したがって、図29Aに示されるように、垂直投影特性を調べて、高いしきい値292より高い上の点で囲まれてい

るような低いしきい値291より下の点が存在するかどうかを判別する。このような条件を満たす点が検出されれば、処理をステップS2803に進める。このような条件を満たす点が検出されなければ、以下に述べるようにしきい値の変更を行う。

【0146】まず、低いしきい値291を垂直投影特性の最大値の10%とし、高いしきい値292を垂直投影特性の最大値の20%とする。ここで、高いしきい値と低いしきい値に関する条件を満たす点が検出されない場合には、高いしきい値と低いしきい値との双方とも垂直投影特性の最大値の2%だけ増加させる。図29Aでは、しきい値291と292の条件を満たす点は検出されない。そこで、しきい値を図29Bに示すように増加させて、低いしきい値291以下であって、高いしきい値292より高い点295と296が両側に存在するような点294を検出する。点294を検出すると、ステップS2803に処理を進め、点294を囲む谷側面の傾きを算出する。谷の右側面の傾きは点294と295とを結ぶ線の傾きであり $\theta_1$ で示される。同様に、谷の左側面の傾きは点294と296とを結ぶ線の傾きであり $\theta_2$ で示される。

【0147】そこで、ステップS2804に処理を進め、角度 $\theta_1$ と $\theta_2$ および角度 $\theta_1$ と $\theta_2$ の近傍で回転させて投影特性を求める。すなわち、回転投影特性を角度 $\theta_1$ 、 $\theta_1 \pm 3^\circ$ 、 $\theta_1 \pm 6^\circ$ 、 $\theta_2$ 、 $\theta_2 \pm 3^\circ$ 、 $\theta_2 \pm 6^\circ$ で求める。この回転投影特性は文字ブロック中の画素を三角関数変換することで求められる。より簡易な手法として、各回転角度（最も近い角度に近似）ごとに画素位置が示される表を用意して、テーブルルックアップで回転投影特性を求めることもできる。なお、回転投影特性の各点は、この画素位置の和として求められる。

【0148】図29Cの297は回転投影特性の典型例を示したものである。回転投影特性297上の各点は、ここでは $\theta_1$ の回転方向の画素数の和として求められる。上述のように、ここでの和は、文字ブロック中の画像を三角関数変換することで求めることもできるし、各回転角度ごとに用意された表を参照して簡易に求めることもできる。

【0149】図29CとDの点線で示されるようなすべての回転投影特性を求めると、ステップS2805に処理を進め、各回転投影特性（10個すべて）ならびにステップS2801で求めた垂直投影特性とを比較して、すべての投影特性の中で最小点を検出する。最小点となる投影特性の角度が切り出し角度に対応する。すなわち、図29Cに示されるように、点299が11個の投影特性の中で最小点となれば、文字ブロックの切り出しが角度 $\theta_1$ で最小点299の位置で行われる（ステップS2806）。

【0150】階層3の切り出し処理を終えると、切り出

された文字に対して認識処理267を行う。ここで、階層3で切り出された2つの文字ブロックとも認識が行われたならば、図2のステップS209に処理を進める。しかし、ここでもなお認識不能なブロックが残る可能性があり、不適切な切り出しが行われた可能性と切り出された文字ブロックを再接続すべきであるという点を考慮する必要がある。この処理は再接続ブロック269で行われ、図30に詳細に示されている。

【0151】ステップS3001では、階層3での切り出し部分の双方に対して267の認識処理を試みる。ステップS3002で2つの要素の認識が可能であると判断されると、上述の図2のステップS209に処理を進める。一方、2つの切り出し要素とも認識不能である場合には、ステップS3003において少なくとも1つの要素が認識可能であるかどうかを判別する。2つの要素とも認識不能である場合にはステップS3004に進み、各要素に対して階層3の切り出し処理とステップS3001などの処理とを行う。

【0152】一方、ステップS3003において少なくとも1つの要素が認識可能であると判断されると、ステップS3005に処理を進め、認識不能の要素に対してさらに階層3の切り出し処理を行う。そして、ステップS3006で新たに切り出された要素が認識可能であると判断されれば、認識不能な要素はなくなり図2のステップS209に処理を進める。これに対して、新たに切り出された要素がともに認識不能であるとステップS3006で判断されると、ステップS3007に処理を進め、不適切な切り出しブロックを再接続することを考える。

【0153】図31は、階層3の処理において不適切な切り出しが行われる可能性を説明する図である。図31Aはイタリック文字「hm」がかなり劣化している例を示したものである。図20の266の階層3の処理では第1の切り出しを301で行うため、文字「h」の垂直部とループ部とが分離してしまう。そして、切り出された各要素に対して267で認識処理を行い、第1のブロックは「l」として認識され、第2のブロックは認識不能と判断されたとする。

【0154】このような場合、処理はステップS3005に進み、認識不能の要素303に対して図31Bのようにさらなる階層3の切り出し処理が行われる。階層3の処理により、304でさらなる切り出しが行われ、切り出された要素305と306に対して267の認識処理が行われる。しかし、要素305は文字として認識不能であるため、再接続処理が必要であると判断される。

【0155】そこで、ステップS3007において、認識不能の切り出し要素を、あらかじめ切り出された隣接する要素と接続する。この際、隣接する要素は認識されたものでも、認識不能のものでも良い。すると、図31Cのように、要素302を要素305と再接続して文字



「h」をほとんど含む新たな要素302'が生成される。そこで、要素302'と要素306とに対して267の認識処理を行う(ステップS3008)。

【0156】これらの処理を終え、ステップS3001に処理を進め、2つの要素ともに認識されたかどうかを調べる。この例の場合には、2つの要素とも文字「h」、「m」として認識されたため処理を終了する。一方、2つの要素とも認識不能であった場合には、上述の処理を繰り返す。

【0157】

【発明の効果】以上説明したように、本発明により、高速かつ正確に文書上の文字を認識してテキストファイルを作成する文字認識方法及び装置を提供できる。また、高速かつ正確に文書上のテキストと非テキストとを選別して、テキストブロックを割り出す方法及び装置を提供できる。

【0158】また、高速かつ正確に文書上の非テキストを分類できる方法及び装置を提供できる。また、高速かつ正確にテキストブロックからテキストラインを分割できる方法及び装置を提供できる。また、高速かつ正確に傾いた文書のテキストブロックからテキストラインを分割できる方法及び装置を提供できる。

【0159】また、高速かつ正確にテキストラインから文字を切り出す方法及び装置を提供できる。また、不適切に切り出された文字の再切り出しが可能な方法及び装置を提供できる。

【図面の簡単な説明】

【図1】本実施例の装置のブロック図である。

【図2】文字認識処理の流れを示すフローチャートである。

【図3】本実施例に基づくブロック分類と選別の処理の流れを示すフローチャートである。

【図4A】画素画像データ中の連結部位の分類処理の流れを示すフローチャートである。

【図4B】画素画像データ中の連結部位の分類処理の流れを示すフローチャートである。

【図4C】画素画像データ中の連結部位の分類処理の流れを示すフローチャートである。

【図5A】輪郭追跡を説明するための図である。

【図5B】輪郭追跡を説明するための図である。

【図5C】輪郭追跡を説明するための図である。

【図6A】非テキスト部位の分類処理を説明するための図である。

【図6B】非テキスト部位の分類処理を説明するための図である。

【図6C】非テキスト部位の分類処理を説明するための図である。

【図7A】白輪郭処理を説明するための図である。

【図7B】白輪郭処理を説明するための図である。

【図7C】白輪郭処理を説明するための図である。

【図7D】白輪郭処理を説明するための図である。

【図8】他のテキスト部位のサイズと近さに基づいてテキスト部分を水平方向に選択的に連結してテキストラインを形成する処理の流れを示すフローチャートである。

【図9】他のテキストラインのサイズと近さに基づいてテキストラインを垂直方向に選択的に連結してテキストブロックを形成する処理の流れを示すフローチャートである。

【図10】代表的な画像の画素データを示す図である。

10 【図11】ブロック分類と選別処理を説明するための図である。

【図12】ブロック分類と選別処理を説明するための図である。

【図13】ブロック分類と選別処理を説明するための図である。

【図14】図11から13にそれぞれ対応する典型的な階層の木構造である。

【図15】図11から13にそれぞれ対応する典型的な階層の木構造である。

20 【図16】図11から13にそれぞれ対応する典型的な階層の木構造である。

【図17】本実施例のライン分割処理の流れを示すフローチャートである。

【図18A】本実施例のライン分割処理を説明するための図である。

【図18B】本実施例のライン分割処理を説明するための図である。

【図18C】本実施例のライン分割処理を説明するための図である。

30 【図18D】本実施例のライン分割処理を説明するための図である。

【図19A】本実施例のライン分割処理を説明するための図である。

【図19B】本実施例のライン分割処理を説明するための図である。

【図19C】本実施例のライン分割処理を説明するための図である。

【図19D】本実施例のライン分割処理を説明するための図である。

40 【図20】本実施例の文字切り出し処理の機能ブロック図である。

【図21】図20の階層1の文字切り出し処理の流れを示すフローチャートである。

【図22】階層1の切り出し処理を説明するための図である。

【図23】図20における単一スペースモード(クエリエフォントなど)の階層2の文字切り出し処理の流れを示すフローチャートである。

50 【図24】階層2の文字切り出し処理を説明するための図である。



【図25】図20における全スペースモード（比例スペースなど）の階層2の文字切り出し処理の流れを示すフローチャートである。

【図26A】階層2の文字切り出し処理を説明するための図である。

【図26B】階層2の文字切り出し処理を説明するための図である。

【図26C】階層2の文字切り出し処理を説明するための図である。

【図26D】階層2の文字切り出し処理を説明するための図である。 10

【図27】階層2の処理で切り出された複数ストローク文字を再融合するための再融合手法を示すフローチャートである。

【図28】図20の階層3の処理の流れを示すフローチャートである。

【図29A】階層3の文字切り出し処理を説明するための図である。

【図29B】階層3の文字切り出し処理を説明するための図である。

【図29C】階層3の文字切り出し処理を説明するための図である。 20

【図29D】階層3の文字切り出し処理を説明するための図である。

【図30】図20の階層3の処理で切り出された部位の再融合処理の流れを示すフローチャートである。

【図31A】再連結処理を説明するための図である。

【図31B】再連結処理を説明するための図である。

【図31C】再連結処理を説明するための図である。

【図32】文字認識すべき文書のページの代表例である。 30

【図33A】従来のライン分割手法を説明するための図である。

【図33B】従来のライン分割手法を説明するための図である。

【図33C】従来のライン分割手法を説明するための図である。

【図34A】従来の文字切り出し手法を説明するための図である。

【図34B】従来の文字切り出し手法を説明するための図である。 40

【図35】本実施例のブロック選別プログラムのソースコードを示す図である。

【図36】本実施例のブロック選別プログラムのソースコードを示す図である。

【図37】本実施例のブロック選別プログラムのソースコードを示す図である。

【図38】本実施例のブロック選別プログラムのソースコードを示す図である。

【図39】本実施例のブロック選別プログラムのソース 50

コードを示す図である。

【図40】本実施例のブロック選別プログラムのソースコードを示す図である。

【図41】本実施例のブロック選別プログラムのソースコードを示す図である。

【図42】本実施例のブロック選別プログラムのソースコードを示す図である。

【図43】本実施例のブロック選別プログラムのソースコードを示す図である。

【図44】本実施例のブロック選別プログラムのソースコードを示す図である。

【図45】本実施例のブロック選別プログラムのソースコードを示す図である。

【図46】本実施例のブロック選別プログラムのソースコードを示す図である。

【図47】本実施例のブロック選別プログラムのソースコードを示す図である。

【図48】本実施例のブロック選別プログラムのソースコードを示す図である。

【図49】本実施例のブロック選別プログラムのソースコードを示す図である。

【図50】本実施例のブロック選別プログラムのソースコードを示す図である。

【図51】本実施例のブロック選別プログラムのソースコードを示す図である。

【図52】本実施例のブロック選別プログラムのソースコードを示す図である。

【図53】本実施例のブロック選別プログラムのソースコードを示す図である。

【図54】本実施例のブロック選別プログラムのソースコードを示す図である。

【図55】本実施例のブロック選別プログラムのソースコードを示す図である。

【図56】本実施例のブロック選別プログラムのソースコードを示す図である。

【図57】本実施例のブロック選別プログラムのソースコードを示す図である。

【図58】本実施例のブロック選別プログラムのソースコードを示す図である。

【図59】本実施例のブロック選別プログラムのソースコードを示す図である。

【図60】本実施例のブロック選別プログラムのソースコードを示す図である。

【図61】本実施例のブロック選別プログラムのソースコードを示す図である。

【図62】本実施例のブロック選別プログラムのソースコードを示す図である。

【図63】本実施例のブロック選別プログラムのソースコードを示す図である。

【図64】本実施例のブロック選別プログラムのソース

コードを示す図である。

【図 90】本実施例のブロック選別プログラムのソースコードを示す図である。

【図 9 1】本実施例のブロック選別プログラムのソースコードを示す図である。

【図 9 2】本実施例のブロック選別プログラムのソースコードを示す図である。

【図 9 3】本実施例のブロック選別プログラムのソースコードを示す図である。

【図94】本実施例のブロック選別プログラムのソースコードを示す図である。

【図 9 5】本実施例のブロック選別プログラムのソースコードを示す図である。

【図 9 6】本実施例のブロック選別プログラムのソースコードを示す図である。

【図 97】本実施例のブロック選別プログラムのソースコードを示す図である。

【図 98】本実施例のブロック選別プログラムのソースコードを示す図である。

【図 9 9】本実施例のブロック選別プログラムのソースコードを示す図である。

【図100】本実施例のブロック選別プログラムのソースコードを示す図である。

【図101】本実施例のブロック選別プログラムのソースコードを示す図である。

【図102】本実施例のブロック選別プログラムのソースコードを示す図である。

【図103】本実施例のブロック選別プログラムのソースコードを示す図である。

【図104】本実施例のブロック選別プログラムのソースコードを示す図である。

【図105】本実施例のブロック選別プログラムのソースコードを示す図である。

【図106】本実施例のブロック選別プログラムのソースコードを示す図である。

【図107】本実施例のブロック選別プログラムのソースコードを示す図である。

【図 108】本実施例のブロック選別プログラムのソースコードを示す図である。

【図１０９】本実施例のブロック選別プログラムのソースコードを示す図である。

【図 110】本実施例のブロック選別プログラムのソースコードを示す図である。

【図 111】本実施例のブロック選別プログラムのソースコードを示す図である。

【図 1 1 2】本実施例のブロック選別プログラムのソースコードを示す図である。

【図 1 1 3】本実施例のブロック選別プログラムのソースコードを示す図である。

【図 114】本実施例のブロック選別プログラムのソー





70

グラムのソースコードを示す図である。

【図 232】本実施例の単一スペース（輪郭）分割プログラムのソースコードを示す図である。

【図 233】本実施例の単一スペース（輪郭）分割プログラムのソースコードを示す図である。

【図 234】本実施例の単一スペース（輪郭）分割プログラムのソースコードを示す図である。

【図 235】本実施例の単一スペース（輪郭）分割プログラムのソースコードを示す図である。

【図 236】本実施例の単一スペース（輪郭）分割プログラムのソースコードを示す図である。

【図 237】本実施例の単一スペース（輪郭）分割プログラムのソースコードを示す図である。

【図 238】本実施例の単一スペース（輪郭）分割プログラムのソースコードを示す図である。

【図 239】本実施例の単一スペース（輪郭）分割プログラムのソースコードを示す図である。

【図 240】本実施例の単一スペース（輪郭）分割プログラムのソースコードを示す図である。

【図 241】本実施例の単一スペース（輪郭）分割プログラムのソースコードを示す図である。

【図 242】本実施例の単一スペース（輪郭）分割プログラムのソースコードを示す図である。

【図 2 4 3】本実施例の単一スペース（輪郭）分割プログラムのソースコードを示す図である。

【図 244】本実施例の単一スペース（輪郭）分割プログラムのソースコードを示す図である。

【図 245】本実施例の単一スペース（輪郭）分割プログラムのソースコードを示す図である。

【図 2 4 6】本実施例の単一スペース（輪郭）分割プログラムのソースコードを示す図である。

【図 247】本実施例の単一スペース（輪郭）分割プログラムのソースコードを示す図である。

【26B】



【图 126】



```
#define FILE_LENGTH      80
#define LINE_LENGTH      256
#define FONT_XMAX_LENGTH 30
```

【例 3 6】



```

#include "pch.h"

int main(argv, argc)
{
    unsigned char **argv;

    argv = popen_blockall();

    //
    // Initialize architecture
    //

    WinPopLevel = Initialize("C:\Nish Selection Demo", argc, argv);

    //
    // Create and popen the first window of the interface.
    //

    sv = popen_blockall();

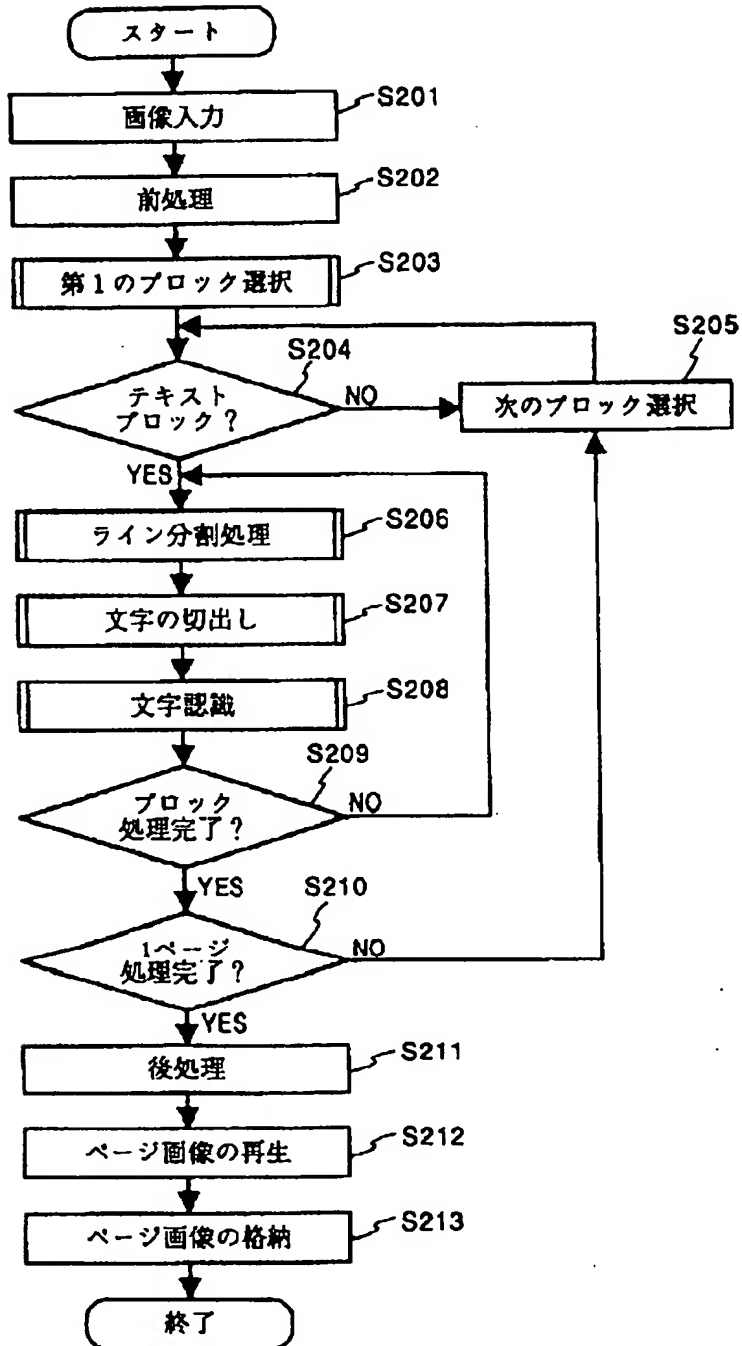
    //
    // Enter the event loop; this function never returns.
    //

    WaitForMsg();

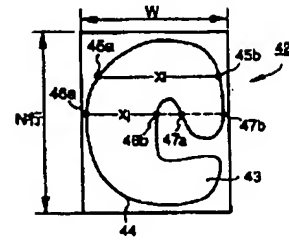
    return 0;
}

```

【図2】

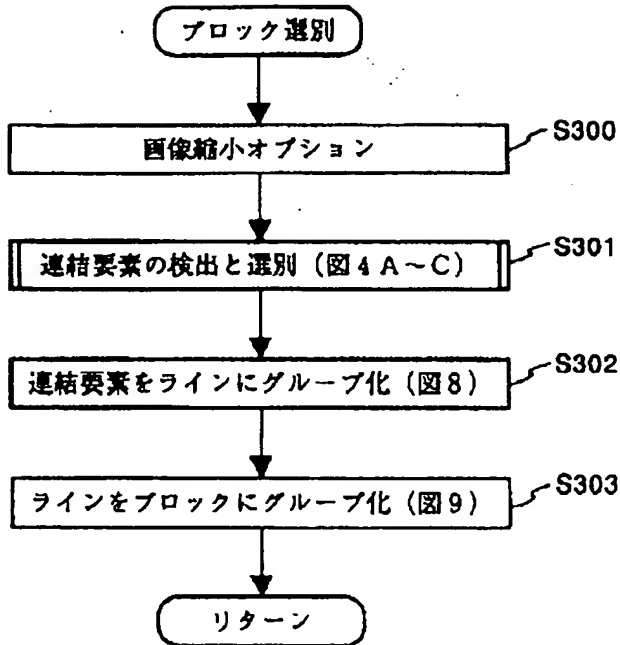


【図6A】

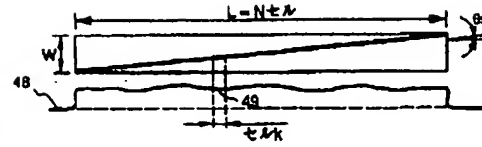




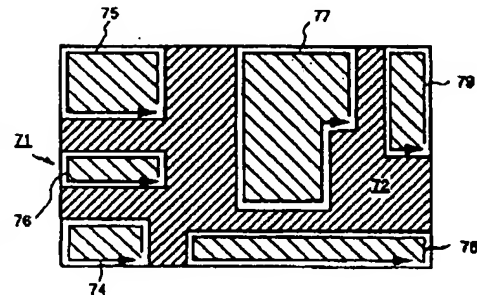
【図3】



【図6B】



【図7B】



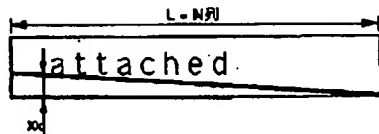
【図194】

```

free(char **M);

```

【図6C】



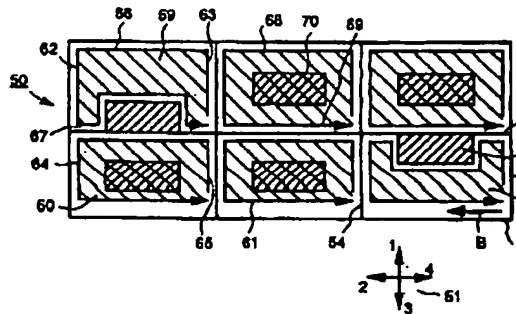
【図234】

```

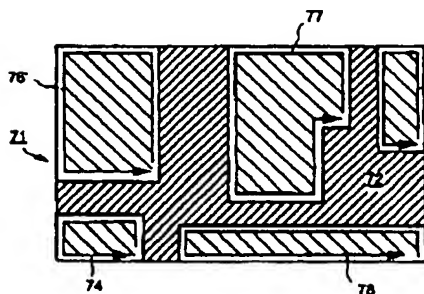
free(char **M);

```

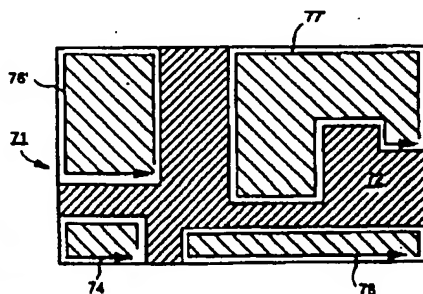
【図7A】



【図7C】



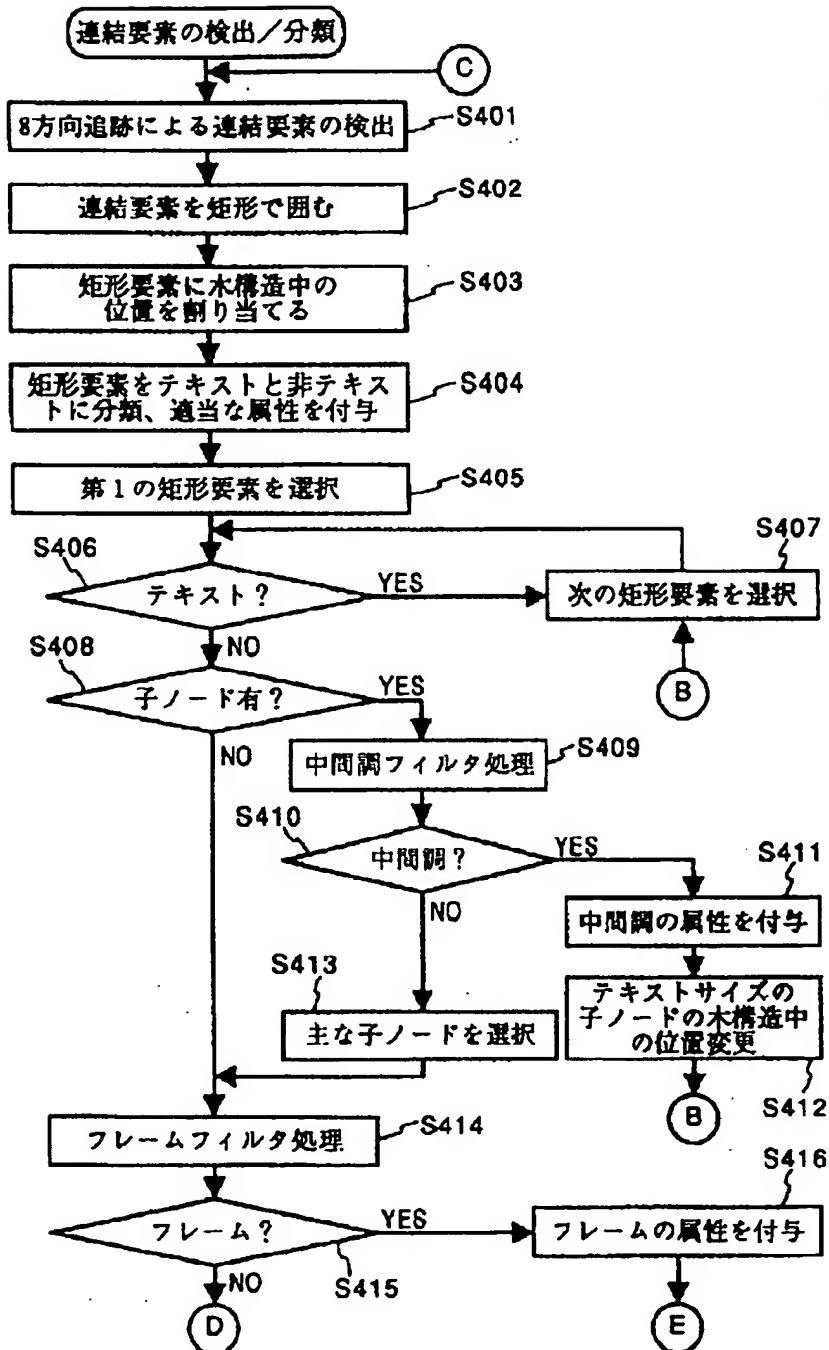
【図7D】



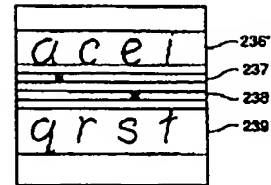
【図26D】



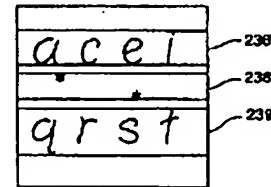
【図4A】



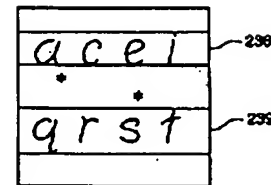
【図18B】



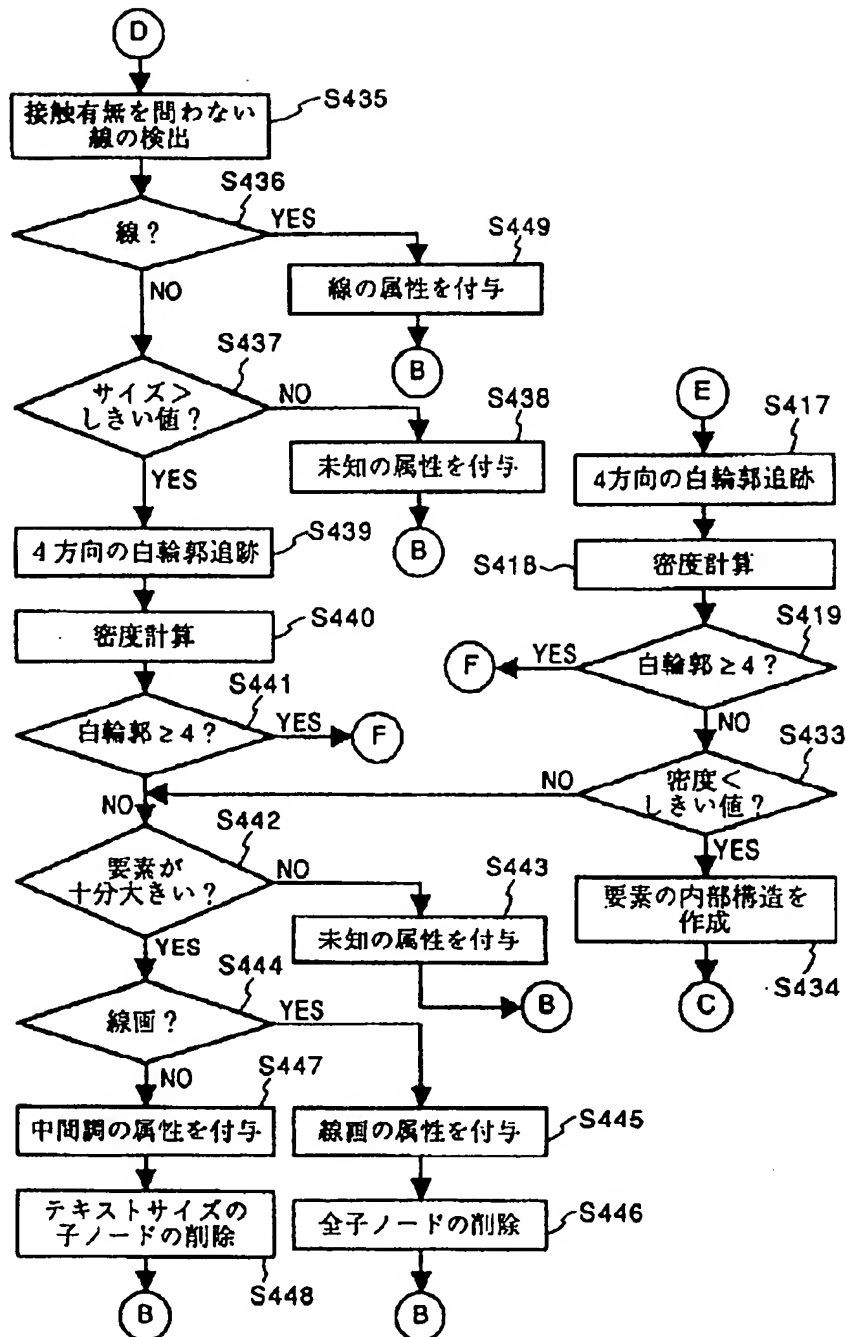
【図18C】



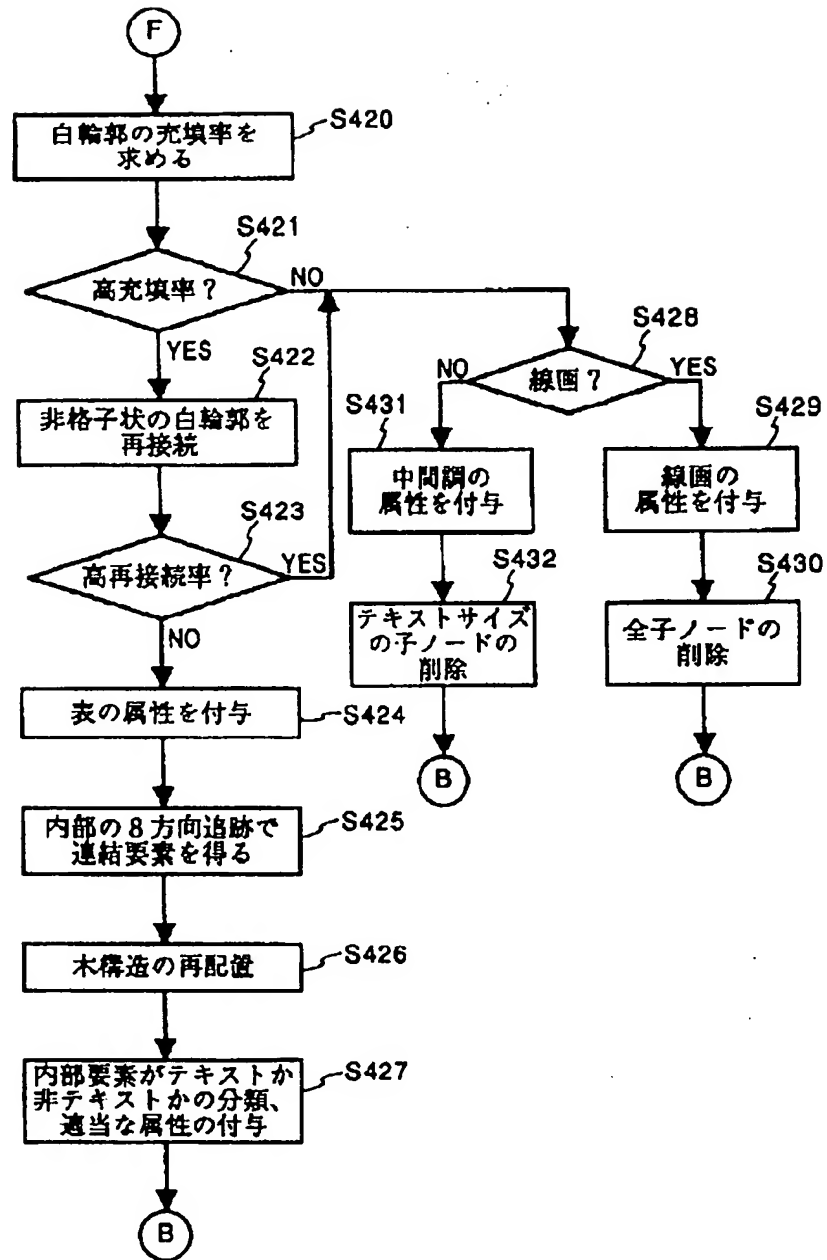
【図18D】



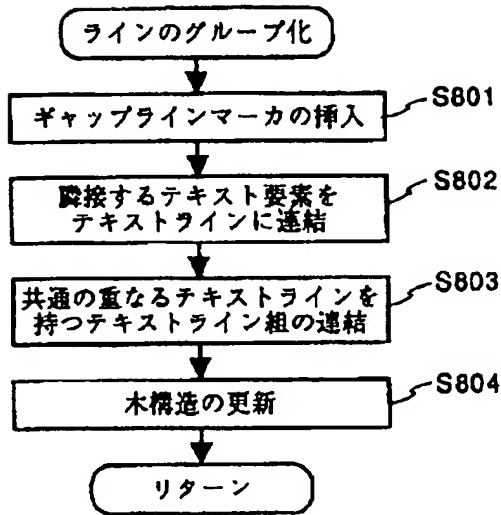
【図4B】



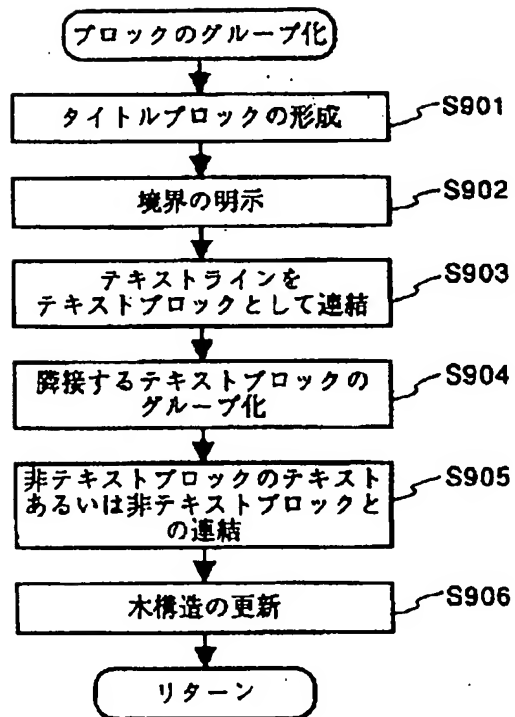
【図4C】



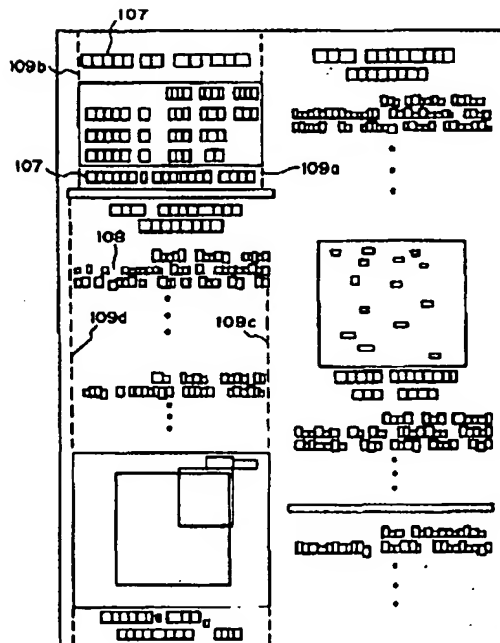
【図8】



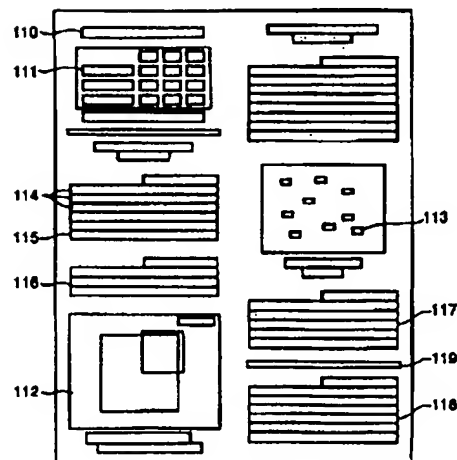
【図9】



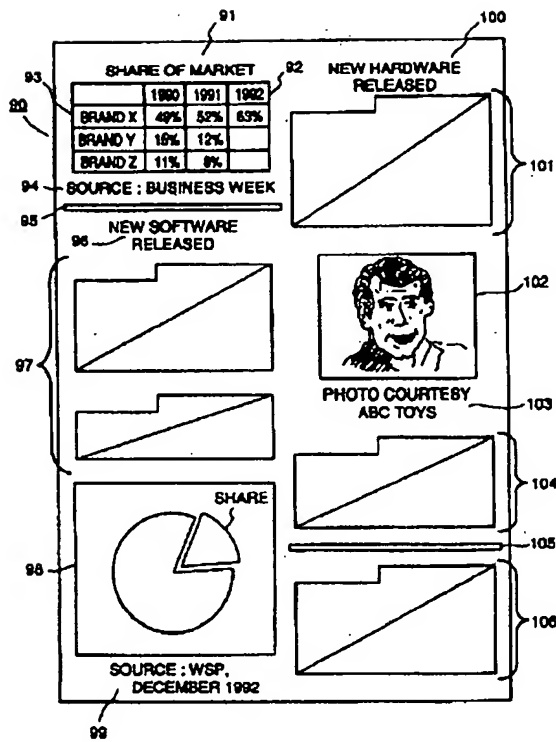
【図11】



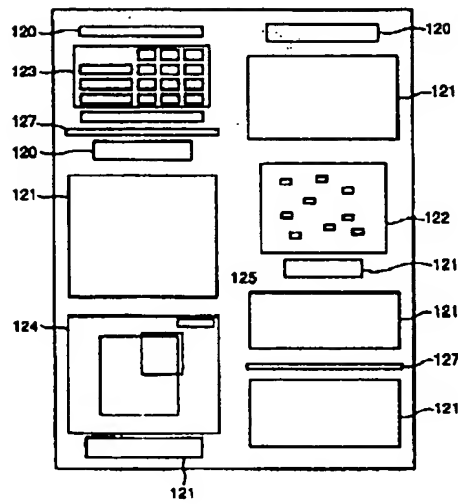
【図12】



【図10】



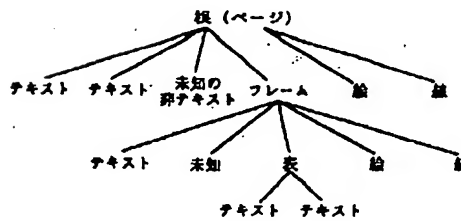
【図13】



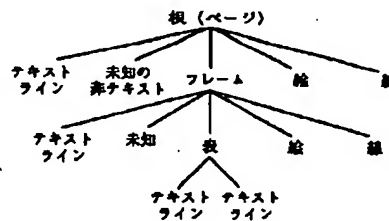
【図26A】

Satisfy

【図14】

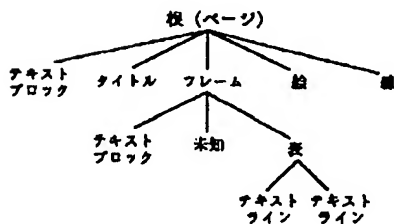


【図15】

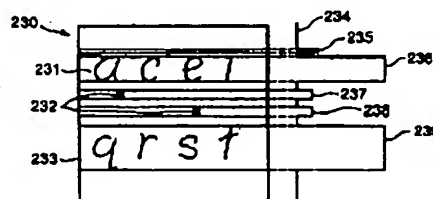


【図35】  
Source Code For  
Block Selection

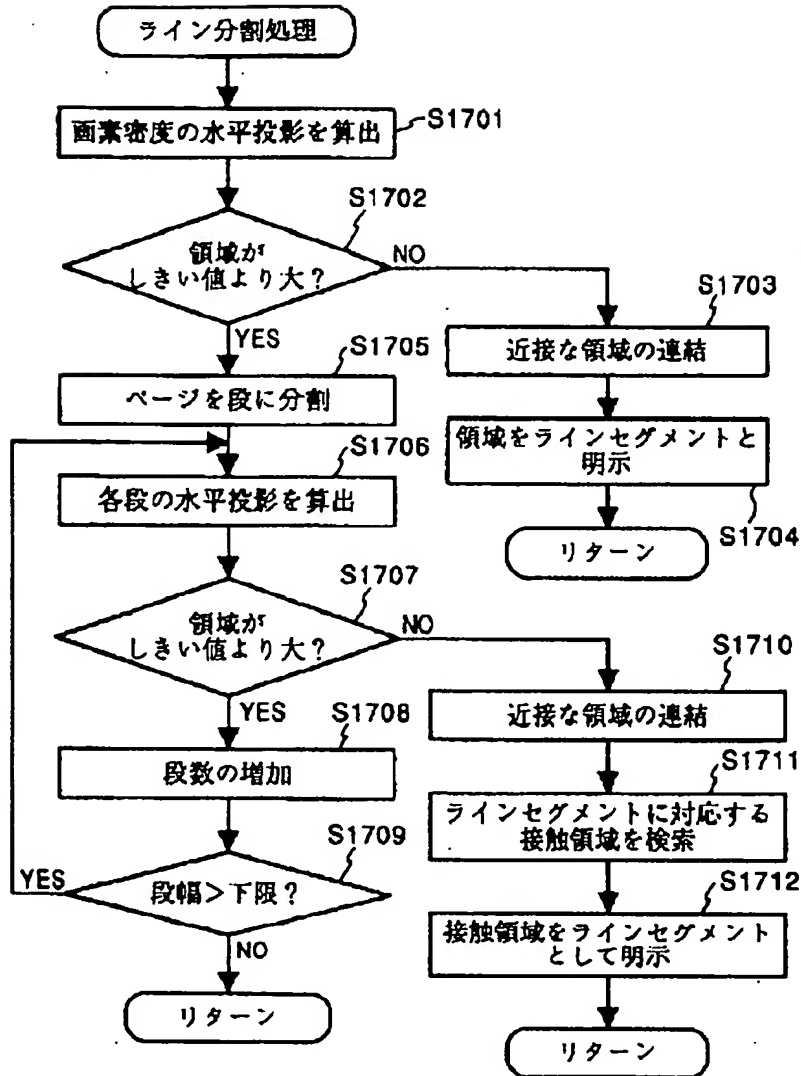
【図16】



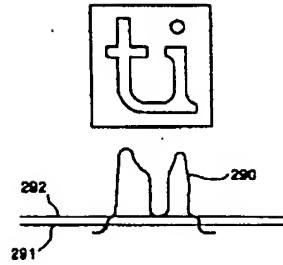
【図18A】



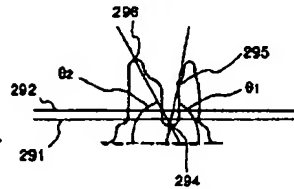
【図17】



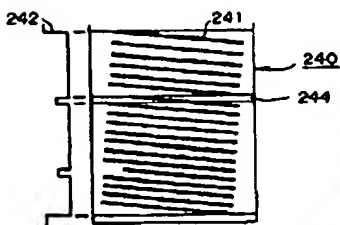
【図29A】



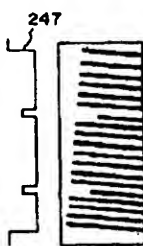
【図29B】



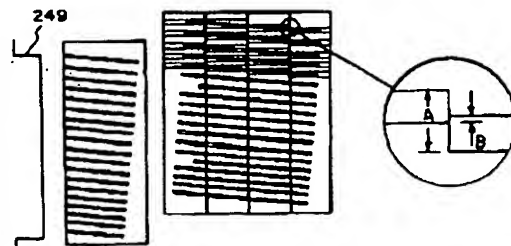
【図19A】



【図19B】

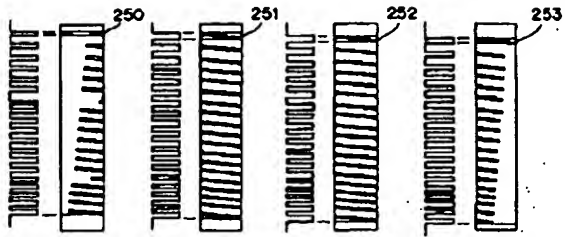


【図19D】

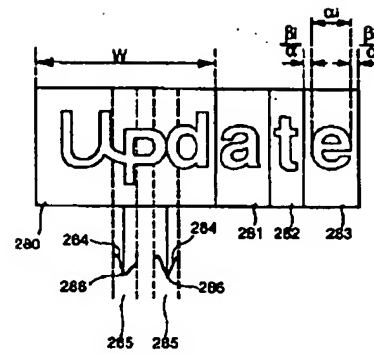




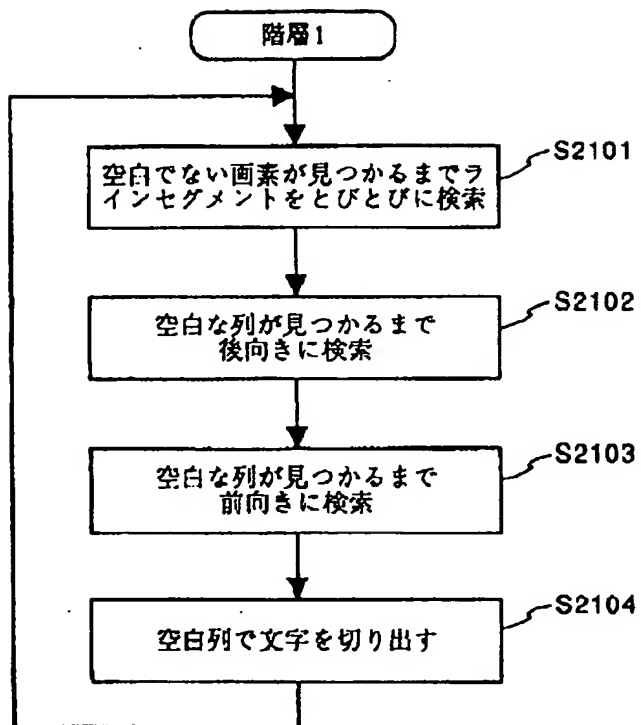
【図19C】



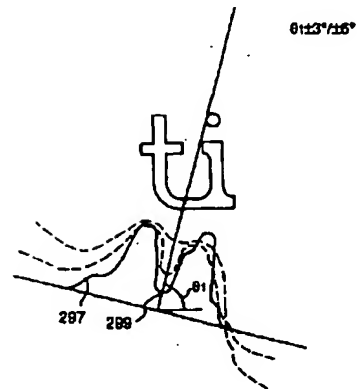
【図24】



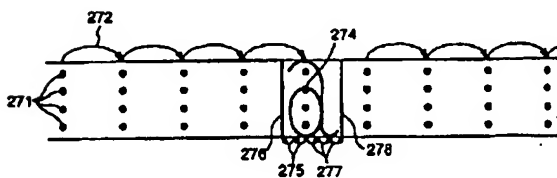
【図21】



【図29C】



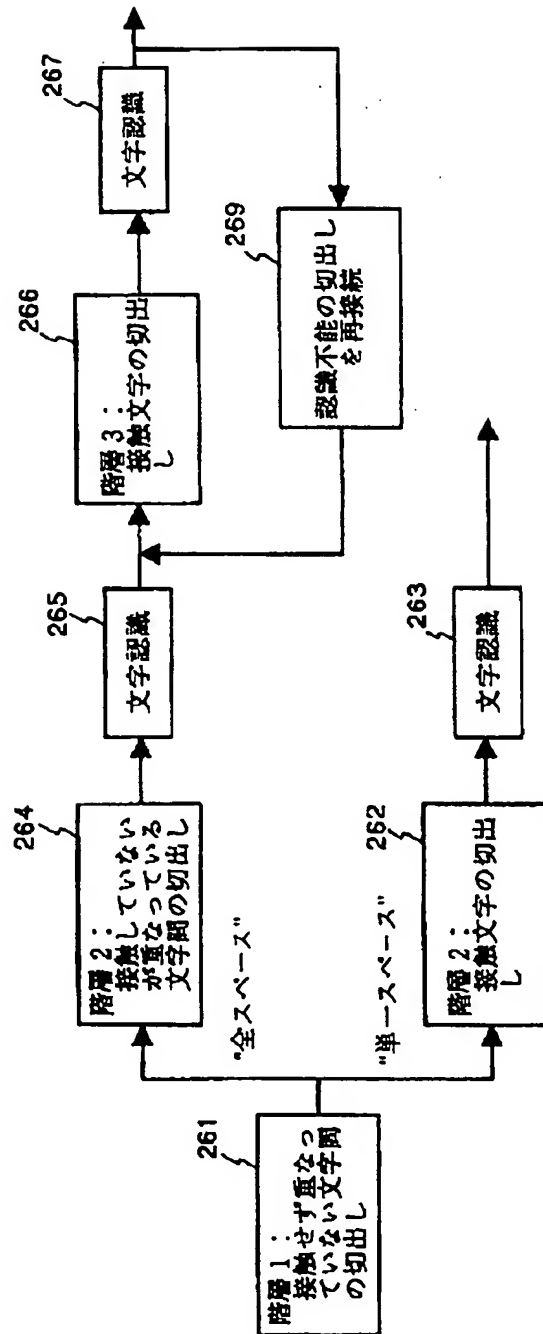
【図22】



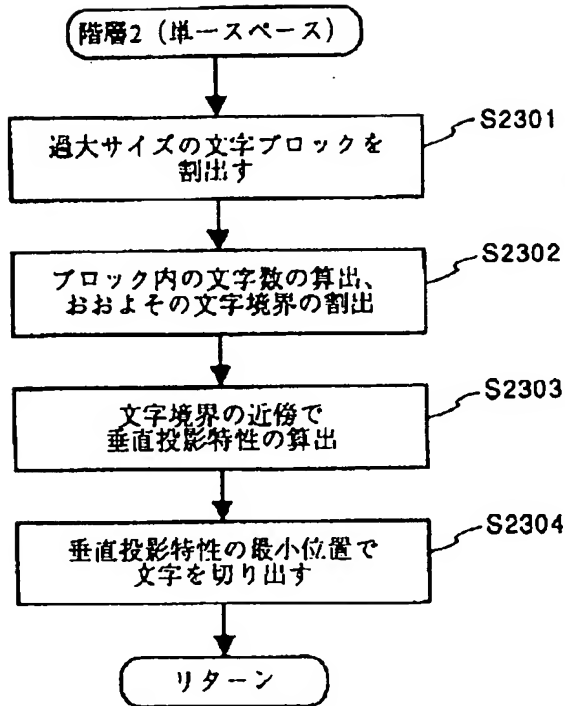
【図29D】



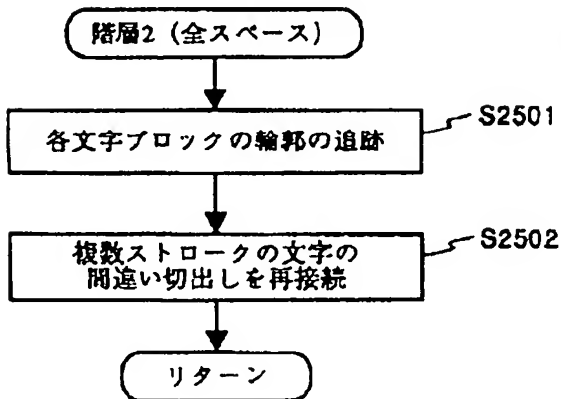
【図20】



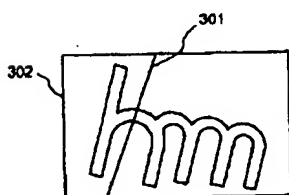
【図23】



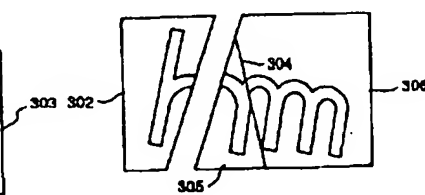
【図25】



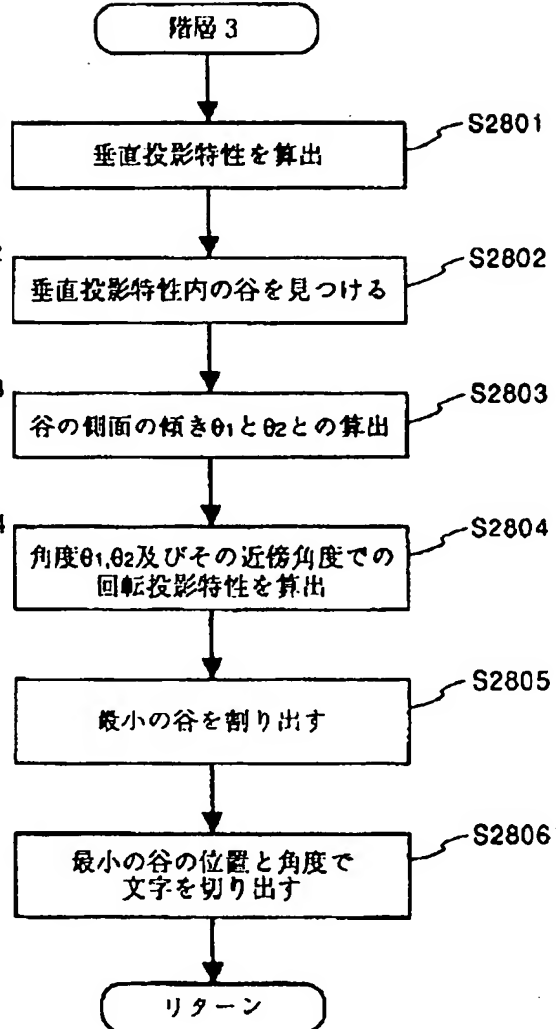
【図31A】



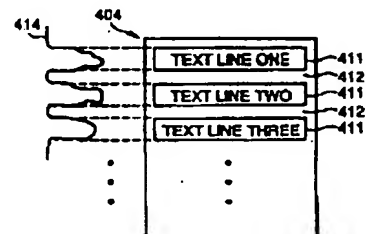
【図31B】



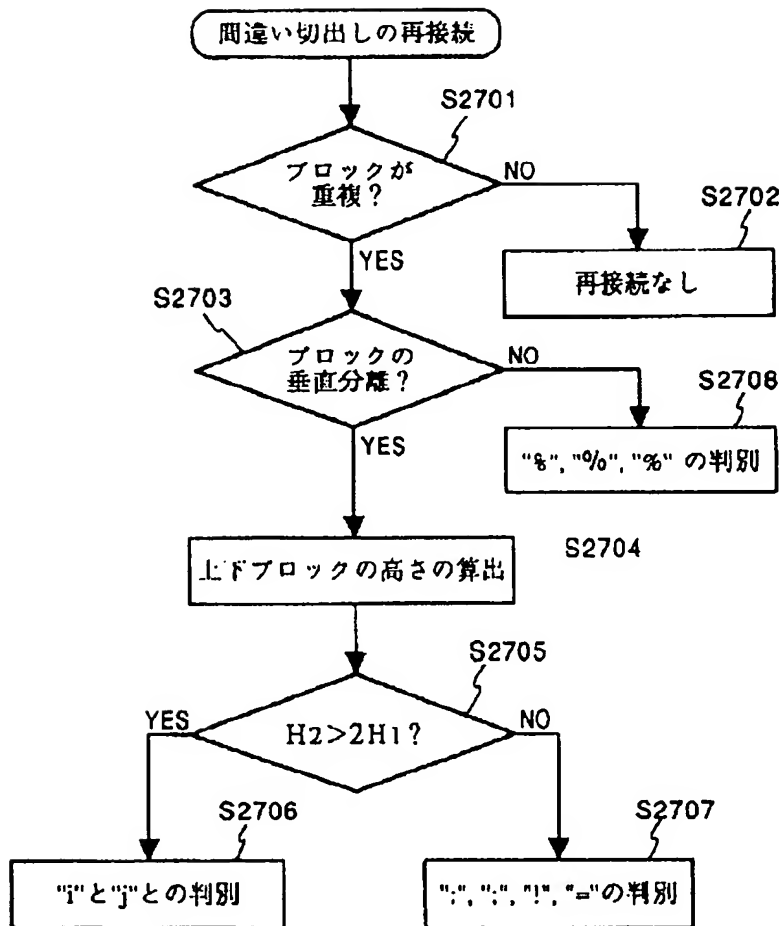
【図28】



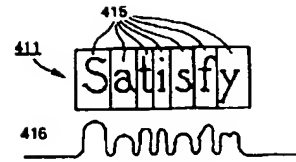
【図33A】



【図27】



【図34A】



【図34B】

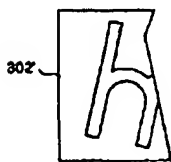


【図134】

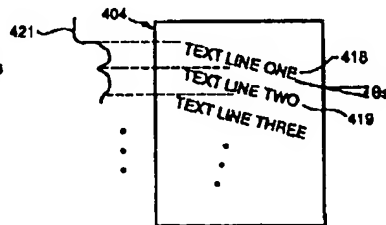
```

extern float *vector();
extern unsigned char *vector();
extern int *ivector();
extern double *dvector();
extern unsigned char *matrix();
extern float *matrix();
extern int **imatrix();
extern double **dmatrix();
extern void free_vector();
extern void free_ivector();
extern void free_dvector();
extern void free_matrix();
extern void free_imatrix();
extern void free_dmatrix();
  
```

【図31C】



【図33B】



【図129】

```

struct sectionblock *previous;
struct sectionblock *next;

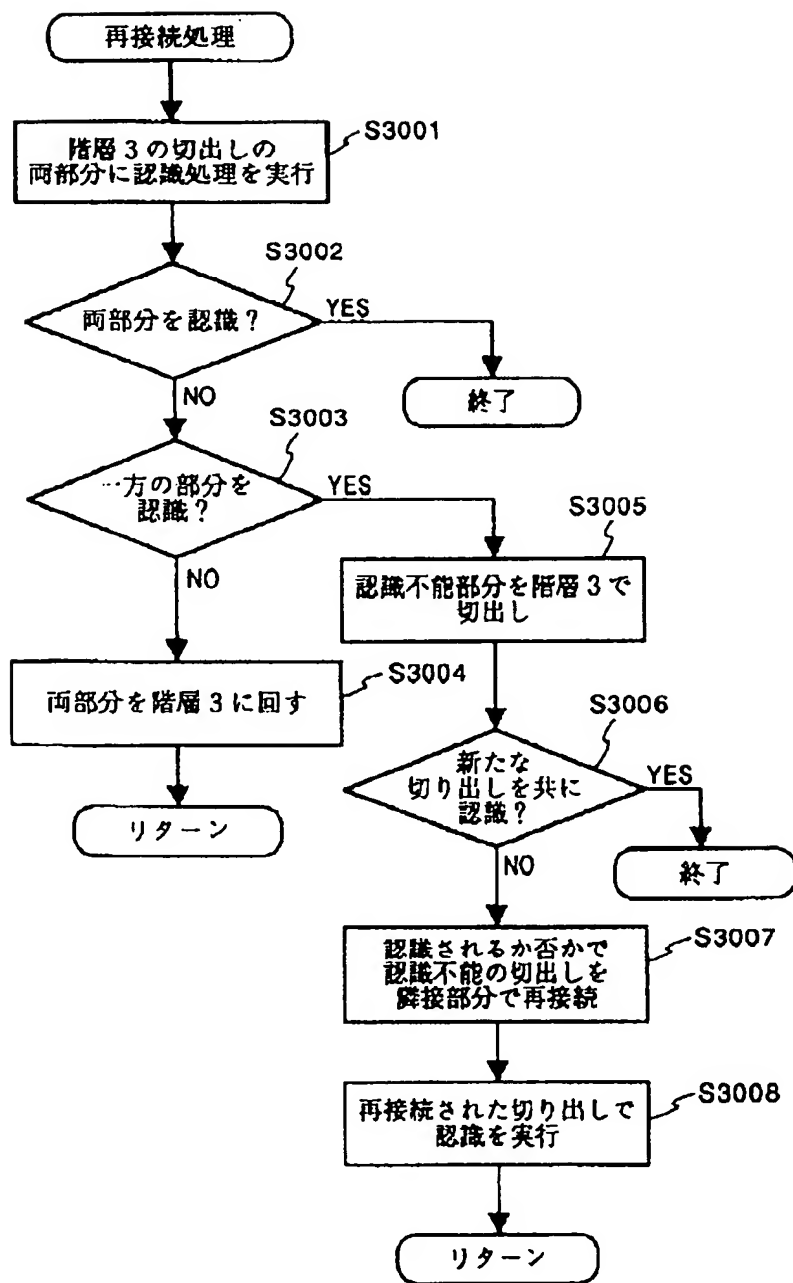
int upper_y, lower_y, left_x, right_x;
int width, length;
int lindex;
int outlineblock;
int ass;
int boundary_lindex;
  
```

【図163】

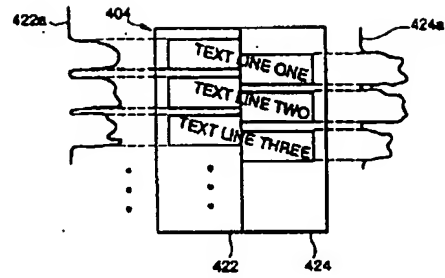
```

#define HISTOGRAM_THRESHOLD
charon = charon + MAX; // Get the next character
free_vector(histogram); // Deallocate the memory
  
```

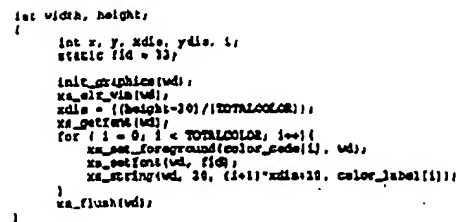
【図30】



【图 3 3 C】



【图 6 2】



【☒ 1 3 3】

[illegible]

【图 4-6】

[illegible]













【図42】

```

        _UnCon->FileSelectionDialog= FileSelectionDialog;
        _UnCon->FileSelectionBox= FileSelectionBox;
    }
    _UnCon->parent= parent;

static void _UnFromContext(_UnCon)
    _UnCFileSelectionDialog *_UnCon;
{
    FileSelectionDialog= _UnCon->FileSelectionDialog;
    FileSelectionBox= _UnCon->FileSelectionBox;
    parent= _UnCon->parent;
}

/* ARGSUSED */
static void cancelCallback_FileSelectionBox(UxWidget, UxContext, UxCallbackArg)
    Widget      UxWidget;
    _UnCFileSelectionDialog *_UnContext;
    int          UxCallbackArg;
{
    UxWidget      UxThisWidget;
    UxThisWidget= UxWidgetToWidget(UxWidget);
    _UnFromContext(UxContext);

    {
        UxPopdownInterface(FileSelectionDialog);
    }
    _UnToContext(UxContext);
}

/* ARGSUSED */
static void okCallback_FileSelectionBox(UxWidget, UxContext, UxCallbackArg)
    Widget      UxWidget;
    _UnCFileSelectionDialog *_UnContext;
    int          UxCallbackArg;
{
    UxWidget      UxThisWidget;
    UxThisWidget= UxWidgetToWidget(UxWidget);
    _UnFromContext(UxContext);

    {
        handle_tiff_handle;
        UxPopdownInterface(FileSelectionDialog);
        strcpy(filename, out_input(filename));
        printf("Open (%s)\n", filename);
        tiff_handle = Uxtreatasubproc ("user/local/bin/X11/imageview", filename,
UxWidget);
        if (tiff_handle == -1) {
            error_create_subproc ();
            return;
        }
        if (UxRunSubproc (tiff_handle, NULL) == -1) {
            error_run_subproc ();
            return;
        }
        open_tiff_file(filename);
    }
    _UnToContext(UxContext);
}

```

【図48】

```

}

static void _UnFromContext(_UnCon)
    _UnCInfoForm *_UnCon;
{
    InfoForm= _UnCon->InfoForm;
    InfoFormClose_PBO= _UnCon->InfoFormClose_PBO;
    InfoFormArea= _UnCon->InfoFormArea;
}

/* ARGSUSED */
static void act(va;callback_InfoFormClose_PBO(UxWidget, UxContext, UxCallbackArg)
    Widget      UxWidget;
    _UnCInfoForm *_UnContext;
    int          UxCallbackArg;
{
    UxWidget      UxThisWidget;
    UxThisWidget= UxWidgetToWidget(UxWidget);
    _UnFromContext(UxContext);

    {
        UxPopdownInterface(InfoForm);
    }
    _UnToContext(UxContext);
}

```

【图 4-3】

[illegible]

—59—

[illegible]





【図49】

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>

struct timeval { first, second, third;
struct timeval tip;

/*.....*/
/* test the timing clock
start_clock ()
{
    gettimeofday (&first, &second);
}

/*.....*/
/* stop the timing clock and return the elapsed time in milliseconds.
double stop_clock ()
{
    double time;
    gettimeofday (&second, &third);
    if (first.tv_usec > second.tv_usec)
        second.tv_usec -= 1000000;
    else
        second.tv_usec += first.tv_usec;
    time = (double)((double)(second.tv_usec - first.tv_usec) / 1000);
    printf ("*** Elapsed time is %.2f\n msec.\n", time);
    return time;
}

/*.....*/
/* Return the basename of a file without the directory ->
char *extract_basename(char *name)
{
    char *name;
    char tmp[PATH_MAX];
    *pos;
    strcpy (tmp, name);
    if (!pos = strrchr(tmp, '/')) return NULL;
    else
        return pos;
}

/*.....*/
/* Return the basename of a file ->
char *extract_basename(char *name)
{
    char *name;
    char tmp[PATH_MAX];
    *pos;
    strcpy (tmp, name);
    if (!pos = strrchr(tmp, '/')) return NULL;
    else
        return pos;
}

/*.....*/
/* Return the basename of a file ->
char *extract_basename(char *name)
{
    char *name;
    char tmp[PATH_MAX];
    *pos;
    strcpy (tmp, name);
    if (!pos = strrchr(tmp, '/')) return NULL;
    else
        return pos;
}

/*.....*/
/* Return the basename of a file ->
char *extract_basename(char *name)
{
    char *name;
    char tmp[PATH_MAX];
    *pos;
    strcpy (tmp, name);
    if (!pos = strrchr(tmp, '/')) return NULL;
    else
        return pos;
}

```



—63—

[illegible]





【図54】

【図217】

## Source Code for

Mono-Spaced ("Courier")  
Segmentation

```

if (v == EOF)
{
    printf("\n Error: Premature End of File found in file (line)",
        mid(i));
}
if ((v & 128) == black*128)
    page->pixels[i][j] = 0;
else
    page->pixels[i][j] = 1;
v = v << 1;
}
}
fclose(tagfile);
return page;
}

```

【図86】

```

        release_block(tmp1);
        tmp1 = tmp11;
    }
    for (i = tmp1; i != NULL; i++)
    {
        tmp11 = tmp1->next;
        release_block(tmp1);
        tmp1 = tmp11;
    }
    free_rectblock(initial_block);
    return;
}
/* ..... Draw text components for display ..... */
/* ..... */
void block_construct(p, width, height, p_ori)
/* signed char **p;
   int width, height;
   unsigned char **p_ori;
   {
    int i, j;
    /* open file for storing block data */
    original_p = p_ori;
    reduce_factor = SCALE_RATIO;

    for (i = 0; i <= 8; i++)
        for (j = 0; j <= 8; j++)
            direct_status[i][j] = 0;

    block_file = fopen("block_file.dat", "w");

    text_length = 150.0/SCALE_RATIO;
    total_text = 0;

    /* the root block stands for the whole page image unit */
    root_rectblock = new_block(0,0);
    root_rectblock->width = width;
    root_rectblock->length = height;
    root_rectblock->upper_y = 0;
    root_rectblock->lower_y = height-1;
    root_rectblock->left_x = 0;
    root_rectblock->right_x = width-1;

    /* find blocks inside the page range */
    block_in_range(p, 0, 0, width-1, height-1, root_rectblock);
    printf("get out of block_in_range\n");
    rearrange_rectblock(root_rectblock);
    printf("get out of rearrange_rectblock\n");

    text_discriminate(root_rectblock);

    printf("get out of text_disc\n");

    firstlevelmontextsearch(p, root_rectblock);

    printf("get out of (firstlevelmontextsearch)\n");
    print_block(1, root_rectblock);

    (close(block_file));
    printf("before get out of block_construct\n");
}

```

【图 5-5】

[illegible]





【例 5 7】

[illegible]

【例 58】

[illegible]

[図59]

```

    cap1 = NULL;
    cap2 = cap1;
}
/* Print free section cap1 = id 'n', cap1;
*/
free_section_block *cap1;
cap1 = NULL;
cap2 = cap1;
}
for (cap1 = first_section; cap1 != NULL; ) {
    print_free_section_block(cap1);
    for (cap2 = cap1->next; cap2 != NULL; ) {
        print_free_section_block(cap2);
        cap2 = cap2->next;
    }
    cap1 = cap1->next;
}
/* Print free line cap1 = id 'n', cap1;
*/
free_line_block *cap1;
cap1 = NULL;
cap2 = cap1;
}
for (cap1 = first_line_block; cap1 != NULL; ) {
    print_free_line_block(cap1);
    for (cap2 = cap1->next; cap2 != NULL; ) {
        print_free_line_block(cap2);
        cap2 = cap2->next;
    }
    cap1 = cap1->next;
}
/* Print free block cap1 = id 'n', cap1;
*/
free_block *cap1;
cap1 = NULL;
cap2 = cap1;
}
for (cap1 = first_block; cap1 != NULL; ) {
    print_free_block(cap1);
    for (cap2 = cap1->next; cap2 != NULL; ) {
        print_free_block(cap2);
        cap2 = cap2->next;
    }
    cap1 = cap1->next;
}
/* Print free table cap1 = id 'n', cap1;
*/
free_table *cap1;
cap1 = NULL;
cap2 = cap1;
}
for (cap1 = first_table; cap1 != NULL; ) {
    print_free_table(cap1);
    for (cap2 = cap1->next; cap2 != NULL; ) {
        print_free_table(cap2);
        cap2 = cap2->next;
    }
    cap1 = cap1->next;
}
/* Print free image cap1 = id 'n', cap1;
*/
free_image *cap1;
cap1 = NULL;
cap2 = cap1;
}
for (cap1 = first_image; cap1 != NULL; ) {
    print_free_image(cap1);
    for (cap2 = cap1->next; cap2 != NULL; ) {
        print_free_image(cap2);
        cap2 = cap2->next;
    }
    cap1 = cap1->next;
}
/* Print free font cap1 = id 'n', cap1;
*/
free_font *cap1;
cap1 = NULL;
cap2 = cap1;
}
for (cap1 = first_font; cap1 != NULL; ) {
    print_free_font(cap1);
    for (cap2 = cap1->next; cap2 != NULL; ) {
        print_free_font(cap2);
        cap2 = cap2->next;
    }
    cap1 = cap1->next;
}
/* Print free color cap1 = id 'n', cap1;
*/
free_color *cap1;
cap1 = NULL;
cap2 = cap1;
}
for (cap1 = first_color; cap1 != NULL; ) {
    print_free_color(cap1);
    for (cap2 = cap1->next; cap2 != NULL; ) {
        print_free_color(cap2);
        cap2 = cap2->next;
    }
    cap1 = cap1->next;
}
/* Print free text cap1 = id 'n', cap1;
*/
free_text *cap1;
cap1 = NULL;
cap2 = cap1;
}
for (cap1 = first_text; cap1 != NULL; ) {
    print_free_text(cap1);
    for (cap2 = cap1->next; cap2 != NULL; ) {
        print_free_text(cap2);
        cap2 = cap2->next;
    }
    cap1 = cap1->next;
}
/* Print free table cap1 = id 'n', cap1;
*/
free_table *cap1;
cap1 = NULL;
cap2 = cap1;
}
for (cap1 = first_table; cap1 != NULL; ) {
    print_free_table(cap1);
    for (cap2 = cap1->next; cap2 != NULL; ) {
        print_free_table(cap2);
        cap2 = cap2->next;
    }
    cap1 = cap1->next;
}
/* Print free image cap1 = id 'n', cap1;
*/
free_image *cap1;
cap1 = NULL;
cap2 = cap1;
}
for (cap1 = first_image; cap1 != NULL; ) {
    print_free_image(cap1);
    for (cap2 = cap1->next; cap2 != NULL; ) {
        print_free_image(cap2);
        cap2 = cap2->next;
    }
    cap1 = cap1->next;
}
/* Print free font cap1 = id 'n', cap1;
*/
free_font *cap1;
cap1 = NULL;
cap2 = cap1;
}
for (cap1 = first_font; cap1 != NULL; ) {
    print_free_font(cap1);
    for (cap2 = cap1->next; cap2 != NULL; ) {
        print_free_font(cap2);
        cap2 = cap2->next;
    }
    cap1 = cap1->next;
}
/* Print free color cap1 = id 'n', cap1;
*/
free_color *cap1;
cap1 = NULL;
cap2 = cap1;
}
for (cap1 = first_color; cap1 != NULL; ) {
    print_free_color(cap1);
    for (cap2 = cap1->next; cap2 != NULL; ) {
        print_free_color(cap2);
        cap2 = cap2->next;
    }
    cap1 = cap1->next;
}
/* Print free text cap1 = id 'n', cap1;
*/
free_text *cap1;
cap1 = NULL;
cap2 = cap1;
}
for (cap1 = first_text; cap1 != NULL; ) {
    print_free_text(cap1);
    for (cap2 = cap1->next; cap2 != NULL; ) {
        print_free_text(cap2);
        cap2 = cap2->next;
    }
    cap1 = cap1->next;
}

```

【图 60】

[illegible]



【图 6-3】

[illegible]

【図64】

```

/* Allocate a double matrix with range [n1..n2] [n1..n2] */
double **matrix(n1, n2, n1, n2)
{
    int n1, n2, n1, n2;
    int i, j;
    double **m;
    m = (double **) malloc((n2-n1+1)*sizeof(double));
    if (!m) perror("allocation failure 1 in matrix");
    m += n1;
    for (i = n1; i <= n2; i++)
        m[i] = (double *) malloc((n2-n1+1)*sizeof(double));
        if (!m[i]) perror("allocation failure 2 in matrix");
        m[i] += n1;
    return m;
}

/* deallocate a float vector */
void free_vector(float *v, n1, n2)
{
    int n1, n2;
    free((float *)v);
}

/* deallocate a int vector */
void free_vector(int *v, n1, n2)
{
    int n1, n2;
    free((int *)v);
}

/* deallocate a double vector */
void free_vector(double *v, n1, n2)
{
    int n1, n2;
    free((double *)v);
}

/* deallocate a float matrix */
void free_matrix(float **m, n1, n2, n1, n2)
{
    int n1, n2, n1, n2;
    int i;
    for (i = n1; i <= n2; i++) free((float *) m[i]);
    free((float **) m);
}

/* deallocate an int matrix */
void free_matrix(int **m, n1, n2, n1, n2)
{
    int n1, n2, n1, n2;
    int i;
    for (i = n1; i <= n2; i++) free((int *) m[i]);
    free((int **) m);
}

/* deallocate a char matrix */
void free_matrix(char **m, n1, n2, n1, n2)
{
    int n1, n2, n1, n2;
    int i;
    for (i = n1; i <= n2; i++) free((char *) m[i]);
    free((char **) m);
}

/* deallocate a double matrix */
void free_matrix(double **m, n1, n2, n1, n2)
{
    int n1, n2, n1, n2;
    int i;
    for (i = n1; i <= n2; i++) free((double *) m[i]);
    free((double **) m);
}

```





—77—

```

struct data_outline *new_data_outline(nl, nh)
{
    int nl, nh;

    struct data_outline *v;
    struct data_outline *v;
    v = (struct data_outline *) malloc((unassigned(nh-nl)+1)
    sizeof(struct data_outline));
    if (!v) {
        printf("allocation error: data_outline\n");
        exit(1);
    }
    v = v+n;
    for (i = nl; i < nh; i++) {
        v[i].total_left = v[i].total_right = 0;
        v[i].current_left = v[i].current_right = NULL;
        v[i].head_left = v[i].head_right = NULL;
    }
    return v;
}

/*..... Allocate memory for _outline */
struct _outline *new_outline(target)
{
    struct _outline *v;

    v = (struct _outline *) malloc(sizeof(struct _outline));
    if (!v) {
        printf("allocation error: _outline\n");
        exit(1);
    }
    v = v+target;
    v->parent = NULL;
    return v;
}

/*..... Search the outline of surrounding "one" */
int test_one_membership(x, y, needisset, xl, yl, x2, y2)
{
    int xl, yl, x2, y2;
    int k, j, count, count;

    k = search_one_membership(x, y, needisset, xl, yl, x2, y2);
    for (j = k; j <= MAXY; j++) {
        count = 0;
        if (search_one_membership(x, y, needisset, xl, yl, x2, y2)) {
            count++;
        }
        if (count > 0) {
            return k;
        }
    }
    return 0;
}

/*..... Adding one element to data_outline on right side */
}

```

—78—

```

.....
void add_right(int, Y, Ylist)
{
    int n, y;
    struct data_node *ylist;

    struct data_node *tmp;

    tmp = new data_node();
    if (ylist != NULL)
        ylist->current_right = tmp;
    ylist->current_right = tmp;
    ylist->total_right++;
}

.....
void add_left(int, Y, Ylist)
{
    int n, y;
    struct data_node *ylist;

    struct data_node *tmp;

    tmp = new data_node();
    if (ylist != NULL)
        ylist->current_left = tmp;
    ylist->current_left = tmp;
    ylist->total_left++;
}

.....
void add_element(int, Y, Ylist)
{
    int n, y;
    struct data_node *ylist;

    struct data_node *tmp;

    tmp = new data_node();
    if (ylist != NULL)
        ylist->current_node = tmp;
    ylist->current_node = tmp;
    ylist->total_node++;
}

.....
void add_element(int, Y, location, Ylist)
{
    struct data_node *ylist;

    struct data_node *tmp;

    if (location == LEFT)
        add_left(n, y, Ylist);
    else if (location == RIGHT)
        add_right(n, y, Ylist);
    else if (location == MIDDLE)
        add_element(n, y, Ylist);
}

.....
void bubble_sort(Ylist, int, int)
{
    struct data_node *ylist;

    struct data_node *tmp;

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n-i-1; j++)
        {
            if (ylist[j].data > ylist[j+1].data)
            {
                swap(ylist[j], ylist[j+1]);
            }
        }
    }
}

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040

```





【图 7-1】

[illegible]





[illegible]



```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040

```

—87—

[illegible]

—88—

```

}
else if (avg_zero == 0)
{
    black_density = 0;
    black_group = 0;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < m; j++)
        {
            if (abs(block[i][j] - avg_zero) > 0.5)
            {
                black_group++;
                black_density++;
            }
        }
    }
    black_group = black_group / n;
    black_density = black_density / m;
}
else
{
    black_group = 0;
    black_density = 0;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < m; j++)
        {
            if (abs(block[i][j] - avg_zero) > 0.5)
            {
                black_group++;
                black_density++;
            }
        }
    }
    black_group = black_group / n;
    black_density = black_density / m;
}
}

// Print the results
printf("Black group: %f\n", black_group);
printf("Black density: %f\n", black_density);
}

// End of program
return 0;
}

```

【图 7 8】

[illegible]

【图 7 9】

[illegible]







[図82]

```

        struct block_rect *tmp, *tmpnext;
        float
            tmp1 = (head->avg_height == 0.0) ? cont_length : head->avg_height;
        for (tmp = first->firstsubblock; tmp->NULL; ) {
            if (tmp->cover == COVERED)
                if (!first|tmp->length == tmp1-TEXT_SCALE)
                    tmpnext = tmp->next; at(tmp->first->head->width-width-1);
                else {
                    transfer_block(TEXT, TEXT, tmp, first, head);
                    tmp = tmpnext;
                }
            else {
                if (tmp->length > tmp1-TEXT_SCALE)
                    if (tmp->first|first, tmp, tmp1) continue;
                    tmpnext = tmp->next;
                    transfer_block(TEXT, NONTEXT, tmp, first, head);
                    tmp = tmpnext;
                }
            }
        }
    }
}

/*
 * Recover the covered content block into original head block
 */
void main_recover(head, first)
struct block_rect *head, *first;
{
    struct block_rect *tmp, *tmpnext;
    float
        tmp1 = (head->avg_height == 0.0) ? cont_length : head->avg_height;
    for (tmp = first->firstsubblock; tmp->NULL; ) {
        if (tmp->cover == COVERED)
            tmpnext = tmp->next;
            if (!first|tmp->length == tmp1-TEXT_SCALE)
                transfer_block(TEXT, TEXT, tmp, first, head);
            else {
                if (tmp->length > tmp1-TEXT_SCALE)
                    if (tmp->first|first, tmp, tmp1) continue;
                    tmpnext = tmp->next;
                    transfer_block(TEXT, NONTEXT, tmp, first, head);
                    tmp = tmpnext;
                }
            }
        }
    }
}

void main_recover(head, first)
{
    /*
     * Recover the covered content block into original head block
     */
    void main_recover(head, first);
    for (tmp = first->firstsubblock; tmp->NULL; ) {
        if (tmp->cover == COVERED)
            tmpnext = tmp->next;
            if (!first|tmp->length == tmp1-TEXT_SCALE)
                transfer_block(TEXT, TEXT, tmp, first, head);
            else {
                if (tmp->length > tmp1-TEXT_SCALE)
                    if (tmp->first|first, tmp, tmp1) continue;
                    tmpnext = tmp->next;
                    transfer_block(TEXT, NONTEXT, tmp, first, head);
                    tmp = tmpnext;
                }
            }
        }
    }
}

void main_recover(head, first)
{
    /*
     * Recover the covered content block into original head block
     */
    void main_recover(head, first);
    for (tmp = first->firstsubblock; tmp->NULL; ) {
        if (tmp->cover == COVERED)
            tmpnext = tmp->next;
            if (!first|tmp->length == tmp1-TEXT_SCALE)
                transfer_block(TEXT, TEXT, tmp, first, head);
            else {
                if (tmp->length > tmp1-TEXT_SCALE)
                    if (tmp->first|first, tmp, tmp1) continue;
                    tmpnext = tmp->next;
                    transfer_block(TEXT, NONTEXT, tmp, first, head);
                    tmp = tmpnext;
                }
            }
        }
    }
}

```

【例 8 3】

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

—95—

[illegible]



—97—

[illegible]

【図88】

```

if (first_block == second_block) {
    return 0;
} else {
    continue;
}

} else {
    break;
}

return 0;
}

/* Allocate new line block */
/* ..... */
struct lineblock new_line_block(
    int al, nh,
    struct lineblock *v)
{
    struct lineblock *v;
    v = (struct lineblock *) malloc(sizeof(struct lineblock));
    if (!v) {
        return 0;
    }
    for (i = 0; i < MAX; i++) {
        v[i].current_block = 0;
        v[i].first_lineblock = 0;
        v[i].first_lineblock = 0;
        v[i].previous = v[i].next = 0;
        v[i].width = v[i].height = 0;
        v[i].length = 0;
        v[i].set = 0;
        v[i].max_block = 0;
        v[i].index = 0;
        v[i].sidezip = 0;
        v[i].underception = 0;
        v[i].caption = v[i].caption = 0;
    }
    return v;
}

/* Chain the next line block */
/* ..... */
struct lineblock new_line_block(
    struct lineblock *headline,
    struct lineblock *headline)
{
    struct lineblock *v;
    v = new_line_block(0, 0);
    if (headline == first_lineblock == 0) {
        headline = first_lineblock = v;
    } else {
        headline->next_lineblock = v;
        v->previous = headline;
    }
}

```

【图 8 9】

[illegible]











—104—

[illegible]

—105—

[illegible]

【図96】

```

                                tapline=tapline->next();
for ( i = 0; i <= (level-1)*8; i++)
    fprintf(line_file, " ");
fprintf(line_file, "td content index = %d ", tapline->index);
fprintf(line_file,
"xtd ytd lstd wtd att = %s block indexlstd indexlstd y = %d l = %d r = %d\n",
    tapline->left_x, tapline->upper_y, tapline->length,
    tapline->width, tapline->att, tapline->(first_block->index,
    tapline->current_block->index, tapline->group, tapline->caption,
    tapline->caption));
    print_ln(nextlevel, tapline);
}

/***** Open the file for the output of line message *****/
/***** *****/

void print_lineblock(firstline)
struct lineblock *firstline;
{
    line_file = fopen("line_file.dat", "w");
    print_lnfl, firstline);
    fclose(line_file);
}

```

【図118】

```

                                tapsection = tapsection->next();
if ((tapsection->att&(TITLE|HACKID|VOLUME|REF|FIGURE)) continue;
else
    ini_section_block(tapsection, tapsection->firstlineblock, p,
    column);
}

/***** Print the line information into a file *****/
/***** *****/

void print_sec(level, firstsec)
int level;
struct sectionblock *firstsec;
{
    int nextlevel, i;
    struct sectionblock *tapsubsec, *tapsec;
    nextlevel = level+1;
    for (tapsubsec = firstsec->firstsubsection; tapsubsec != NULL;
        tapsubsec = tapsubsec->next){
        for ( i = 0; i <= (level-1)*8; i++)
            printf(" ");
        printf("index = %d ", tapsubsec->index);
        printf("xtd ytd lstd wtd att = %s\n",
            tapsubsec->left_x, tapsubsec->upper_y,
            tapsubsec->length, tapsubsec->width, tapsubsec->att);
    }
    for (tapsec = firstsec->firstsubsection->next; tapsec != NULL;
        tapsec=tapsec->next){
        for ( i = 0; i <= (level-1)*8; i++)
            printf(" ");
        printf("content index = %d ", tapsec->index);
        printf("xtd ytd lstd wtd att = %s\n",
            tapsec->left_x, tapsec->upper_y, tapsec->length,
            tapsec->width, tapsec->att);
        print_sec(nextlevel, tapsec);
    }

/***** Open the file for the output of line message *****/
/***** *****/

void print_secblock(firstsec)
struct sectionblock *firstsec;
{
    print_sec(1, firstsec);
}

```

【図137】

## Source Code for Proportional Spaced Segmentation

—107—

```

for (i = 0; i < n; i++)
    v[i].column_no = 0;
return v;
}

/*===== allocation section pointers =====*/
/*===== allocation section pointers =====*/

struct sectionblock *new_sectionpointer(nl, nl)
{
    struct sectionblock **v;

    v = (struct sectionblock **) malloc((n+1)*sizeof(struct sectionblock));
    if (!v) {
        printf("error in the allocation of sectionpointer\n");
        exit(1);
    }
    v = v+nl;
    return v;
}

/*===== allocation section pointers =====*/
/*===== allocation section pointers =====*/

void sort_linepointer(line, com)
{
    struct lineblock **line;
    int pos;

    line = lineblock **v;
    struct lineblock *tmp;

    for (i = 0; i < com-1; i++) {
        for (j = 0; stop = 0; j = com - i - 1; j--) {
            if (line[j]-segment_j > line[j+1]-segment_j) {
                tmp = line[j+1];
                line[j+1] = line[j];
                line[j] = tmp;
                stop = 1;
            }
        }
        if (stop == 0) break;
    }
}

/*===== line block sorting =====*/
/*===== line block sorting =====*/

struct lineblock **sort_linepointer(struct lineblock **, struct lineblock **, struct lineblock **)
{
    struct lineblock **v;
    struct lineblock **line;
    int i, j, stop;

    *line = 0;
    for (i = 0; i < firstlineblock-1; i++) {
        *line = *line;
        *line = *line;
    }
    for (i = 0; i < firstlineblock-1; i++) {
        *line = *line;
        *line = *line;
    }
    line = new_linepointer(0, *line);
}

/*===== line block sorting =====*/
/*===== line block sorting =====*/

/*===== allocation new segment structure =====*/
/*===== allocation new segment structure =====*/

struct segment *new_segment(nl, nl)
{
    struct segment **v;
    int i;

    v = (struct segment **) malloc((n+1)*sizeof(struct segment));
    if (!v) {
        printf("error in the allocation of segment\n");
        exit(1);
    }
    v = v+nl;
}

/*===== allocation new segment structure =====*/
/*===== allocation new segment structure =====*/

```



—108—

```

for (i = 0; tmp < firstlineblock->firstsubline; tmp += NULL, tmp = tmp->next)
    link_x[i++] = tmp;

/* ends linking */

if (!firstlineblock->nextsubline || !NULL) {
    /* if firstlineblock->nextsubline is NULL */
    printf("error in entry %d", i);
    link_x[i++] = tmp;

    for (tmp = firstlineblock->firstsubline; tmp != NULL; tmp = tmp->next)
        sort_topinter(link_x, "newline");

    return link_x;
}

/*----- section block sorting -----*/
struct sectionblock *next_section(first, num)
int num;
{
    struct sectionblock *tmp;
    struct sectionblock *link_x;
    int i, j, stop;

    num = 0;
    for (tmp = first->firstsubsection; tmp != NULL; tmp = tmp->next)
        link_x[num++] = tmp;

    link_x = new_sectioninter(0, "num-1");

    for (i = 0; tmp = first->firstsubsection; tmp != NULL; tmp = tmp->next)
        link_x[i++] = tmp;

    for (i = 0; i < (num-1); i++) {
        for (j = 0; stop = 0; j < (num-i-1); j++)
            if ((link_x[j]->supery > link_x[j+1]->supery) ||
                ((link_x[j]->supery == link_x[j+1]->supery) &&
                 (link_x[j]->lower > link_x[j+1]->lower))) {
                tmp = link_x[j+1];
                link_x[j+1] = link_x[j];
                link_x[j] = tmp;
                stop = 1;
            }
        if (stop == 0) break;
    }

    return link_x;
}

/*----- Allocate per line block -----*/
struct sectionblock *new_sectionblock(int, int)
int i;
int num;
{
    struct sectionblock *n;
    v = (struct sectionblock *) malloc(sizeof(struct sectionblock));
    if (!v)
        printf("allocation error: section block %d", i);
    return v;
}

```

```

int stop, tabindex;
struct sameline *tap;
struct lineblock *tap_line;

for ( ; ; ) {
    stop = 0;
    for ( tap = (first-previous) tap != NULL; tap = tap->previous->next; ) {
        tap_line = tap->previous;
        tap->current = tap->previous->next->next;
        tap->index = tap->previous->index;
        tap->previous->index = tap_line;
        tap->previous->next = tap_line;
        stop = 1;
    }
    if (stop == 0) break;
}

/* ===== remove lineblock ===== */
void line_rm(struct, head_line, line_rm)
{
    struct lineblock *head_line, *line_rm;

    if (last == first) {
        if (head_line->next_line == line_rm) {
            head_line->next_line = line_rm->next_line;
            if (line_rm->next == NULL) line_rm->next->previous = NULL;
        } else if (head_line->current->next_line == NULL) {
            head_line->current->next_line = line_rm;
            head_line->current->next_line = line_rm->previous;
            head_line->previous->next = NULL;
        } else {
            line_rm->next->previous = line_rm->previous;
            line_rm->previous->next = line_rm->next;
        }
    } else {
        if (head_line->first->next_line == line_rm) {
            head_line->first->next_line = line_rm->next;
            if (line_rm->next != NULL) line_rm->next->previous = NULL;
        } else if (head_line->current->next_line == NULL) {
            head_line->current->next_line = line_rm;
            head_line->current->next_line = line_rm->previous;
            head_line->previous->next = NULL;
        } else {
            line_rm->next->previous = line_rm->previous;
            line_rm->previous->next = line_rm->next;
        }
    }
}

/* ===== combination of line block forward ===== */
void fill_line_gather(struct, line_rm)
{
    struct lineblock *head_line, *line_rm;
}

if (current->next_lineblock == line) {
    current->first_lineblock = line->next;
    if (current->current_lineblock == line)
        current->current_lineblock->next = NULL;
    else
        current->current_lineblock->next = line;
} else if (current->current_lineblock == line) {
    current->current_lineblock->next = line->next;
    line->previous->next = NULL;
    line->next->previous = line->previous;
    line->next->previous->next = line->next;
} else {
    line->previous = NULL;
    line->next = NULL;
}

void put_lineblock_into_section
{
    struct lineblock *current_section;

    if (current->first_lineblock == NULL) {
        current->first_lineblock = line;
        current->current_lineblock = line;
        current->next_lineblock = line;
        line->next = NULL;
        line->next->previous = line;
    } else {
        current->current_lineblock->next = line;
        line->previous->next = current->current_lineblock;
        line->next->previous = line;
    }
}

void put_lineblock_into_section
{
    struct lineblock *current_section;

    if (current->first_lineblock == NULL) {
        current->first_lineblock = line;
        current->current_lineblock = line;
        current->next_lineblock = line;
        line->next = NULL;
        line->next->previous = line;
    } else {
        current->current_lineblock->next = line;
        line->previous->next = current->current_lineblock;
        line->next->previous = line;
    }
}

/* ===== create new sameline structure ===== */
struct sameline *new_sameline(struct)
{
    struct lineblock *cur;
    struct sameline *s;

    s = (struct sameline *) malloc(sizeof(struct sameline));
    if (!s) printf("allocation error in sameline\n");
    return s;
}

/* ===== sort sameline array by k ===== */
void sort_sameline(struct)
{
    struct sameline *s;
    struct sameline *s1;
    struct sameline *s2;
    struct sameline *s3;
    struct sameline *s4;
    struct sameline *s5;
    struct sameline *s6;
    struct sameline *s7;
    struct sameline *s8;
    struct sameline *s9;
    struct sameline *s10;
    struct sameline *s11;
    struct sameline *s12;
    struct sameline *s13;
    struct sameline *s14;
    struct sameline *s15;
    struct sameline *s16;
    struct sameline *s17;
    struct sameline *s18;
    struct sameline *s19;
    struct sameline *s20;
    struct sameline *s21;
    struct sameline *s22;
    struct sameline *s23;
    struct sameline *s24;
    struct sameline *s25;
    struct sameline *s26;
    struct sameline *s27;
    struct sameline *s28;
    struct sameline *s29;
    struct sameline *s30;
    struct sameline *s31;
    struct sameline *s32;
    struct sameline *s33;
    struct sameline *s34;
    struct sameline *s35;
    struct sameline *s36;
    struct sameline *s37;
    struct sameline *s38;
    struct sameline *s39;
    struct sameline *s40;
    struct sameline *s41;
    struct sameline *s42;
    struct sameline *s43;
    struct sameline *s44;
    struct sameline *s45;
    struct sameline *s46;
    struct sameline *s47;
    struct sameline *s48;
    struct sameline *s49;
    struct sameline *s50;
    struct sameline *s51;
    struct sameline *s52;
    struct sameline *s53;
    struct sameline *s54;
    struct sameline *s55;
    struct sameline *s56;
    struct sameline *s57;
    struct sameline *s58;
    struct sameline *s59;
    struct sameline *s60;
    struct sameline *s61;
    struct sameline *s62;
    struct sameline *s63;
    struct sameline *s64;
    struct sameline *s65;
    struct sameline *s66;
    struct sameline *s67;
    struct sameline *s68;
    struct sameline *s69;
    struct sameline *s70;
    struct sameline *s71;
    struct sameline *s72;
    struct sameline *s73;
    struct sameline *s74;
    struct sameline *s75;
    struct sameline *s76;
    struct sameline *s77;
    struct sameline *s78;
    struct sameline *s79;
    struct sameline *s80;
    struct sameline *s81;
    struct sameline *s82;
    struct sameline *s83;
    struct sameline *s84;
    struct sameline *s85;
    struct sameline *s86;
    struct sameline *s87;
    struct sameline *s88;
    struct sameline *s89;
    struct sameline *s90;
    struct sameline *s91;
    struct sameline *s92;
    struct sameline *s93;
    struct sameline *s94;
    struct sameline *s95;
    struct sameline *s96;
    struct sameline *s97;
    struct sameline *s98;
    struct sameline *s99;
    struct sameline *s100;
    struct sameline *s101;
    struct sameline *s102;
    struct sameline *s103;
    struct sameline *s104;
    struct sameline *s105;
    struct sameline *s106;
    struct sameline *s107;
    struct sameline *s108;
    struct sameline *s109;
    struct sameline *s110;
    struct sameline *s111;
    struct sameline *s112;
    struct sameline *s113;
    struct sameline *s114;
    struct sameline *s115;
    struct sameline *s116;
    struct sameline *s117;
    struct sameline *s118;
    struct sameline *s119;
    struct sameline *s120;
    struct sameline *s121;
    struct sameline *s122;
    struct sameline *s123;
    struct sameline *s124;
    struct sameline *s125;
    struct sameline *s126;
    struct sameline *s127;
    struct sameline *s128;
    struct sameline *s129;
    struct sameline *s130;
    struct sameline *s131;
    struct sameline *s132;
    struct sameline *s133;
    struct sameline *s134;
    struct sameline *s135;
    struct sameline *s136;
    struct sameline *s137;
    struct sameline *s138;
    struct sameline *s139;
    struct sameline *s140;
    struct sameline *s141;
    struct sameline *s142;
    struct sameline *s143;
    struct sameline *s144;
    struct sameline *s145;
    struct sameline *s146;
    struct sameline *s147;
    struct sameline *s148;
    struct sameline *s149;
    struct sameline *s150;
    struct sameline *s151;
    struct sameline *s152;
    struct sameline *s153;
    struct sameline *s154;
    struct sameline *s155;
    struct sameline *s156;
    struct sameline *s157;
    struct sameline *s158;
    struct sameline *s159;
    struct sameline *s160;
    struct sameline *s161;
    struct sameline *s162;
    struct sameline *s163;
    struct sameline *s164;
    struct sameline *s165;
    struct sameline *s166;
    struct sameline *s167;
    struct sameline *s168;
    struct sameline *s169;
    struct sameline *s170;
    struct sameline *s171;
    struct sameline *s172;
    struct sameline *s173;
    struct sameline *s174;
    struct sameline *s175;
    struct sameline *s176;
    struct sameline *s177;
    struct sameline *s178;
    struct sameline *s179;
    struct sameline *s180;
    struct sameline *s181;
    struct sameline *s182;
    struct sameline *s183;
    struct sameline *s184;
    struct sameline *s185;
    struct sameline *s186;
    struct sameline *s187;
    struct sameline *s188;
    struct sameline *s189;
    struct sameline *s190;
    struct sameline *s191;
    struct sameline *s192;
    struct sameline *s193;
    struct sameline *s194;
    struct sameline *s195;
    struct sameline *s196;
    struct sameline *s197;
    struct sameline *s198;
    struct sameline *s199;
    struct sameline *s200;
    struct sameline *s201;
    struct sameline *s202;
    struct sameline *s203;
    struct sameline *s204;
    struct sameline *s205;
    struct sameline *s206;
    struct sameline *s207;
    struct sameline *s208;
    struct sameline *s209;
    struct sameline *s210;
    struct sameline *s211;
    struct sameline *s212;
    struct sameline *s213;
    struct sameline *s214;
    struct sameline *s215;
    struct sameline *s216;
    struct sameline *s217;
    struct sameline *s218;
    struct sameline *s219;
    struct sameline *s220;
    struct sameline *s221;
    struct sameline *s222;
    struct sameline *s223;
    struct sameline *s224;
    struct sameline *s225;
    struct sameline *s226;
    struct sameline *s227;
    struct sameline *s228;
    struct sameline *s229;
    struct sameline *s230;
    struct sameline *s231;
    struct sameline *s232;
    struct sameline *s233;
    struct sameline *s234;
    struct sameline *s235;
    struct sameline *s236;
    struct sameline *s237;
    struct sameline *s238;
    struct sameline *s239;
    struct sameline *s240;
    struct sameline *s241;
    struct sameline *s242;
    struct sameline *s243;
    struct sameline *s244;
    struct sameline *s245;
    struct sameline *s246;
    struct sameline *s247;
    struct sameline *s248;
    struct sameline *s249;
    struct sameline *s250;
    struct sameline *s251;
    struct sameline *s252;
    struct sameline *s253;
    struct sameline *s254;
    struct sameline *s255;
    struct sameline *s256;
    struct sameline *s257;
    struct sameline *s258;
    struct sameline *s259;
    struct sameline *s260;
    struct sameline *s261;
    struct sameline *s262;
    struct sameline *s263;
    struct sameline *s264;
    struct sameline *s265;
    struct sameline *s266;
    struct sameline *s267;
    struct sameline *s268;
    struct sameline *s269;
    struct sameline
```





—112—

[illegible]

—113—

```

1  if (topline->current->right == rightmost)
2      (current-> topline->current->right->
3      line->transfer(topline->current, first, current->current->
4      topline->current->next);
5      current->current->next = first->next;
6      topline->current = NULL;
7      current->right->neighbor(current->line, left->level, right->level);
8      if (rightmost == 1)
9          continue;
10         break;
11     } else {
12         topline->next->connection = NULL;
13         current->current->next = next;
14         break;
15     } else {
16         topline->next->connection = NULL;
17         break;
18     } } else break;
19 }
20 }
21 }
22 }
23 }
24 }
25 }
26 }
27 }
28 }
29 }
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
80
```



-115-

[illegible]





[illegible]

-118-

[illegible]



【図 1 1 0】

[illegible]



【図 1 1 2】

[illegible]

【例 1 1 3】

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```





【例 115】

[illegible]



-127-

[illegible]

【图 1 1 9】

[illegible]

[図120]

```

for (tmp = vtable, num = 0; tmp != NULL; tmp = tmp->next, num++)
    copy[num] = tmp;

for (i = 0; i < num; i++) {
    for (j = 0; j < num; j++) {
        if (copy[j] != white->read_jmp) {
            copy[j+1] = copy[j];
            copy[j] = tmp;
            step = 1;
        }
        if (step == 0) break;
    }
    copy[i] = tmp;
    for (k = 1; k < num; k++) {
        if (copy[k] != white->read_jmp) {
            if (copy[k] != white->read_jmp) {
                copy[k+1] = copy[k];
                copy[k] = tmp;
                step = 1;
            }
            if (step == 0) break;
        }
    }
    if (copy == -1) copy = tmp;
}

if (copy == -1) copy = tmp;

..... first run loop .....
void direct_write(vtable, tmp, copy)
struct white_for_table *vtable;
int *copy, *tmp;

{
    struct white_for_table *tmp2, *tmp;
    int num, step, i, j;

    for (tmp = vtable, num = 0; tmp != NULL; tmp = tmp->next, num++)
        copy = (struct white_for_table *)malloc(num);
    for (tmp = vtable, num = 0; tmp != NULL; tmp = tmp->next, num++)
        copy[num] = tmp;

    for (i = 0; i < num; i++) {
        for (j = 0; j < num; j++) {
            if (copy[j] != white->read_jmp) {
                copy[j+1] = copy[j];
                copy[j] = tmp;
                step = 1;
            }
            if (step == 0) break;
        }
        if (step == 0) break;
    }
    copy[i] = tmp;
    for (k = 1; k < num; k++) {
        if (copy[k] != white->read_jmp) {
            if (copy[k] != white->read_jmp) {
                copy[k+1] = copy[k];
                copy[k] = tmp;
                step = 1;
            }
            if (step == 0) break;
        }
    }
    if (copy == 0) break;
}
copy = tmp;
for (i = 0; i < num; i++) {
    if (copy[i] != white->read_jmp) {
        if (copy[i] != white->read_jmp) {
            copy[i+1] = copy[i];
            copy[i] = tmp;
            step = 1;
        }
        if (step == 0) break;
    }
}
copy = tmp;
}

```

```

..... combine two white .....
void combine_white(vtable, white, white2)
struct white_for_table *vtable;
struct white_for_table *white, *white2;

{
    struct white_for_table *tmpwhite;
    int i, j, total;

    if (white->numwhite == 0) {
        total = white->numwhite;
        if (white->numwhite == 0) total++;
        white->numwhite = new_white(0, total);
        copy_white(white->numwhite, white);
    } else {
        total = white->numwhite + white->numwhite;
        if (white->numwhite == 0) total++;
        tmpwhite = new_white(0, total);
        white->numwhite = new_white(0, total);
        for (i = 0; i < white->numwhite; i++)
            copy_white(white->numwhite(i), tmpwhite(i));
        free(struct white_for_table *tmpwhite);
    }
    if (white->numwhite == 0) {
        copy_white(tmpwhite->numwhite(0), white);
    } else {
        tmpwhite = white->numwhite;
        for (i = 0; i < white->numwhite; i++)
            copy_white(tmpwhite->numwhite(i), tmpwhite(i));
        free(struct white_for_table *tmpwhite);
        white->numwhite = total;
        if (white->numwhite < white->numwhite)
            white->numwhite = white->numwhite;
    }
}

```

【图 121】

[illegible]

[illegible]





[illegible]

【☒ 1 2 5】

```

.....
/* File: block_util.c */
/* Author : Shin-Twan Wang */
/* Date : 1-28-91 */
/* Copyright 1992 Canyon Information Systems */
.....

/* test if covered range overlap */

int if_overlap(a1, a2, b1, b2)
{
    if ((a2 < b1) || (b2 < a1)) return 0;
    return 1;
}

/* minimum of two integers */

int min(a, b)
{
    if (a <= b) return a;
    else return b;
}

/* ratio of overlapping */

float cover_range(a1, a2, b1, b2)
{
    int x1, x2;

    float range = 0.0;

    if ((a2 < b1) || (b2 < a1)) return range;
    else {
        x1 = max(a1, b1);
        x2 = min(a2, b2);
    }

    range = (float)(x2-x1)/(float)(a2-a+1);
    return range;
}

```

【图 131】

[illegible]

【图 1 3 2】

[illegible]



[illegible]

【図135】

```

.....
..... Filename: table.h .....
..... Author: Shin-Yuan Wang .....
..... Date: 1-25-92 .....
..... Copyright 1992 Canon Information Systems .....
.....
#include "blockgeneral.h"

#define NONE 0
#define OFFSET 3
#define OFFSET 4
#define XDM 1
#define OLD 3
#define MONITOR 0
#define STOP 1
#define TABLE_SETTING 1

struct white_for_table
{
    int status;
    struct whitecolor *white;
    struct white_for_table *previous, *next;
};

```

【図136】

```

.....
..... Filename: image.h .....
..... Author: Shin-Yuan Wang .....
..... Date: 1-25-92 .....
..... Copyright 1992 Canon Information Systems .....
.....
#include<stdio.h>
#include<stdlib.h>

struct image
{
    int size;
    int size;
    int dpi;
    unsigned char *pixel;
};

```

【図142】

```

/*
 * Filename: autoinput.h
 * Header file for autoinput.c
 */
.....
/* Function Definitions for mymalloc.c */
#ifdef CPP
/* Definition for C++ files */
extern "C" {
    void auto_initialize();
    int auto_input_int();
    void auto_terminate();
}
#else
/* Definition for C files */
void auto_initialize();
int auto_input_int();
void auto_terminate();
#endif

```

【図145】

```

.....
void barovpp_terminate(mallist)
    struct MoniterovppList *mallist; /* The list */
{
    free_vector(mallist->list); /* Deallocate memory */
    mallist->allocated = 0; /* Set so won't be accidentally used */
    mallist->size = 0;
}

.....
void barovpp_add(mallist, a, b)
    struct MoniterovppList *mallist; /* The list */
    int a, b; /* Start and stop of the barovpp */
{
    if (mallist->size == mallist->allocated) {
        printf("MoniterovppList FULL!!\n");
        exit(1);
    }
    mallist->list[mallist->size++] = a;
    mallist->list[mallist->size++] = b;
}

```

【図244】

```

colors[x].flag = DoRed : DoGreen : DoBlue;

colors[x].red = the_color;
colors[x].green = the_color;
colors[x].blue = the_color;
the_color += 50;
}
XQueryColors(d, del_colormap, colors, ncolors);
my_cmap = XCreateColormap(d, DefaultRootWindow(d), DefaultVisual(d, scr),
                          AllocAll);
XStoreColors(d, my_cmap, colors, ncolors);
};

```

【图 1 3 8】

[illegible]



【図139】

```

// If (a == 3)
// It's evenly aligned
or = oldline.pixel;
nr = newline.pixel;

for (l=0; l<newr; l++)
{
    ocl = *(or++);
    ocl3 = *(or++);
    ocl3 = *(or++);
    nr = *(nr++);
    nr = *(nr++);
    nr = *(nr++);
    *(nr++) = *(ocl++);
    *(nr++) = *(ocl3++);
}

// Do the ends
if (a == 3)
{
    ocl = *or;
    for (j=0; j<newr; j++)
        *(nr++) = *(ocl++);
}
else if (a == 2)
{
    ocl = *(or++);
    ocl3 = *or;
    for (j=0; j<newr; j++)
        *(nr++) = *(ocl3++);
}
}

// And x1 on logic
void clip_x1_x2_31(clipages oldline, clipages newline)
{
    int older, older; // Size of old image
    int newer, newer; // Size of new image
    unsigned char *ocl, *ocl3, *ocl; // The old & new column pointers
    unsigned char *nr, *nr; // The old & new row pointers
    newline.realocate(newer, newer);
    a = older - newer; // How much is partially covered in the last row
    if (a == 3)
        // It's evenly aligned
    or = oldline.pixel;
    nr = newline.pixel;

    for (l=0; l<newr; l++)
    {
        ocl = *(or++);
        ocl3 = *(or++);
        ocl3 = *(or++);
        nr = *(nr++);
        nr = *(nr++);
        nr = *(nr++);
        *(nr++) = *(ocl++);
        *(nr++) = *(ocl3++);
    }

    // Do the ends
    if (a == 3)
    {
        ocl = *or;
        for (j=0; j<newr; j++)
            *(nr++) = *(ocl++);
    }
    else if (a == 2)
    {
        ocl = *(or++);
        ocl3 = *or;
        for (j=0; j<newr; j++)
            *(nr++) = *(ocl3++);
    }
}

// x1 on logic
void clip_x1_x2_31(clipages oldline, clipages newline)
{
    int older, older; // Size of old image
    int newer, newer; // Size of new image
    unsigned char *ocl, *ocl3, *ocl; // The old & new column pointers
    unsigned char *nr, *nr; // The old & new row pointers
    older = oldline.pixel;
    older = oldline.pixel;
    newer = (older-1)/2; // Should be --
    newline.realocate(newer, newer);
    a = older - newer; // How much is partially covered in the last row
    if (a == 3)
        // It's evenly aligned
    or = oldline.pixel;
    nr = newline.pixel;

    for (l=0; l<newr; l++)
    {
        ocl = *(or++);
        ocl3 = *(or++);
        ocl3 = *(or++);
        nr = *(nr++);
        nr = *(nr++);
        nr = *(nr++);
        *(nr++) = *(ocl++);
        *(nr++) = *(ocl3++);
    }

    // Do the ends
    if (a == 3)
    {
        ocl = *or;
        for (j=0; j<newr; j++)
            *(nr++) = *(ocl++);
    }
    else if (a == 2)
    {
        ocl = *(or++);
        ocl3 = *or;
        for (j=0; j<newr; j++)
            *(nr++) = *(ocl3++);
    }
}

// x1 on logic
void clip_x1_x2_31(clipages oldline, clipages newline)
{
    int older, older; // Size of old image
    int newer, newer; // Size of new image
    unsigned char *ocl, *ocl3, *ocl; // The old & new column pointers
    unsigned char *nr, *nr; // The old & new row pointers
    older = oldline.pixel;
    older = oldline.pixel;
    newer = (older-1)/2; // Should be --
    newline.realocate(newer, newer);
    a = older - newer; // How much is partially covered in the last row
    if (a == 3)
        // It's evenly aligned
    or = oldline.pixel;
    nr = newline.pixel;

    for (l=0; l<newr; l++)
    {
        ocl = *(or++);
        ocl3 = *(or++);
        ocl3 = *(or++);
        nr = *(nr++);
        nr = *(nr++);
        nr = *(nr++);
        *(nr++) = *(ocl++);
        *(nr++) = *(ocl3++);
    }

    // Do the ends
    if (a == 3)
    {
        ocl = *or;
        for (j=0; j<newr; j++)
            *(nr++) = *(ocl++);
    }
    else if (a == 2)
    {
        ocl = *(or++);
        ocl3 = *or;
        for (j=0; j<newr; j++)
            *(nr++) = *(ocl3++);
    }
}

```

【图 140】

[illegible]

【図141】

```

/* filename: autoinput.h
 * This file contains the code to input from a filename
 * .....
 */
#include <stdio.h>
#include <stdlib.h>
/* Global variables */
int auto_eof_detected,
FILE *auto_fp;
int save_count;
int auto_eof_detected;
char filename[100];
.....
void auto_initialize()
{
    int not_eof = 1;
    auto_eof_detected = 0;
    while(not_eof)
    {
        printf("Input filename for auto-input: ");
        scanf("%s", filename);
        auto_fp = fopen(filename, "r");
        if (auto_fp == NULL) /* Read open failed */
        {
            auto_fp = fopen(filename, "w");
            if (auto_fp == NULL) /* Write open failed */
            {
                printf("Can't open file %s\n", filename);
                printf("Enter 1 at the next prompt.\n");
                save_count = 1;
                auto_eof_detected = 1;
                not_eof = 0;
            }
        }
        else {
            printf("Opening existing file %s\n", filename);
            not_eof = 0;
        }
    }
    printf("Do you wish to save the auto-input into this file? (1 = yes) ");
    save_count = auto_input_int();
    printf("Enter 1 at the next prompt.\n");
    save_count = 0;
}

int auto_input_int()
{
    int i;
    if (auto_eof_detected)
    {
        printf("\n(Warning) Input (-) = newline, -9999 = exit) : ");
    }
    scanf("%d", &i);
    if (i == -9999)
    {
        fclose(auto_fp);
        exit(0);
    }
    if (save_count == 1)
    {
        /* Print new lines */
        {
            printf(auto_fp, "\n"); /* Print the new lines */
            printf(auto_fp, "\n");
            fflush(auto_fp);
            save_count = 0;
        }
        return auto_input_int(); /* And input again */
    }
    printf(auto_fp, "%d ", i);
    fflush(auto_fp);
    if (save_count > 20)
    {
        save_count = 0;
        printf(auto_fp, "\n");
    }
    return i;
}

if (feof(auto_fp, "r", &i) == EOF)
{
    printf("..... END OF FILE DETECTED .....");
    {
        if (save_count == 1)
        {
            fclose(auto_fp);
            printf("..... WARNING: CANNOT APPEND TO FILE. Is ..... filename,
            if (auto_fp == NULL)
            {
                printf("..... WARNING: CANNOT APPEND TO FILE. Is ..... filename,
                save_count = 0;
                printf(auto_fp, "\n");
            }
            auto_eof_detected = 1;
            return auto_input_int();
        }
        printf(".....");
        return i;
    }
}

```

[ 1 4 3 ]

```

/* filename: charlist.c
 * This file contains Charlist and helper/utility routines
 */
.....

/*-----
 * By Christopher Sherlock
 * Includes 'charlist.h'
 * Includes 'math.h'
 */
.....

/* Initialize and allocate a Charlist (set up curve pointers) */
/* Call this routine once only! */
void Charlist_initialize(Charlist *clist, vector *heights)
{
    int i;
    int clist_max; /* The max number of characters */
    int vector_height; /* Height of the line (its allocate curves) */

    line = 1; /* A looping var */

    /* Allocate memory for characters */
    clist_max = (struct Character *) malloc((unsigned) clist_max
    sizeof(struct Character));
    if (clist_max == 0)
        printf("Memory Allocation Error in Charlist_initialize()\n");
        exit(1);

    /* Number of characters */
    clist_max = 0;
    clist_max_allocated = clist_max; /* Size Allocated */
    clist_first = 0; /* No first character */
    clist_vector_height = vector_height;

    /* Allocate memory for curves and set pointers */
    for (i=0; i<clist_max; i++) {
        clist_max[i].curve = (vector *) malloc(vector_height);
        clist_max[i].curve = (vector *) malloc(vector_height);
    }

    /* Here a previously allocated Charlist */
    void Charlist_initialize(Charlist *clist)
    {
        struct Character *clist;
        clist_max = 0; /* Number of characters */
        clist_first = 0; /* No first character */
    }

    /* Terminates and deallocates the Charlist */
    void Charlist_terminate(Charlist *clist)
    {
        struct Character *clist;
        int i;

        for (i=clist_max-1; i>=0; i--) { /* Free curves */
            free(vector_height[i].curve);
            free(vector_height[i].curve);
        }
    }
}

```

```

}

from(struct Character *) clist_max;
clist_max = 0;
clist_max_allocated = 0;
clist_first = 0;

}

/* Place a new character in the Charlist */
/* Take the character beyond the Charlist (charlist),
void Charlist_initialize(Charlist *clist)
{
    struct Character *clist;
    struct Character *newchar; /* pointer to the new character adding */
    newchar = malloc(sizeof(struct Character)); /* Set newchar */
    /* Calculate the position of the new character */
    Charlist_initialize(clist, newchar);

    /* Check for full list */
    /* If the size of the list is full,
    clist_max++;
    if (clist_max == clist_max_allocated) /* If too big... */
        printf("Error in Charlist_initialize(): Character list is FULL!\n");
        exit(1);
}

/* Place new in the list character */
Charlist_initialize(Charlist *clist, newchar);

/* This routine repositions a character in the Charlist. It removes
the character, recalculates the position and inserts it in the list accordingly
void Charlist_relocate(Charlist *clist, int
struct Character *clist;
struct Character *newchar;

/* Recalculate the position of the character */
Charlist_initialize(Charlist *clist);
if (clist_max < 1) /* If it's the only item in the list */
    return;

/* Remove the character from the list */
if (clist_max == 0) {
    clist_max = clist_max - 1; /* If list on list */
    clist_max_allocated = clist_max_allocated - 1;
} else {
    clist_max = clist_max - 1;
    if (clist_max == 0)
        clist_max_allocated = clist_max_allocated - 1;
}

/* Place the character in the list */
Charlist_initialize(Charlist *clist, newchar);

}

/* Insert a character after doing a exit */
}

```

【図 1 4 4】

```

/* This routine will insert the character beyond the charlist (charlist).
 * loc is not used to place the new character. However,
 * it is used to place the new character. It will
 * place the new character after charlist. After finished, it is an
 * extremely good idea to reset the result!
 */
void CharlistInsertAfterCharlist(charlist, ch)
{
    struct Charlist *charlist; /* The character */
    struct Charlist *charlist; /* The character */
    int loc;
    struct Charlist *charlist; /* Used for searching for location */
    struct Charlist *charlist; /* The character before p */
    void CharlistInsertAfterCharlist(charlist, ch)
    {
        struct Charlist *charlist; /* The character */
        struct Charlist *charlist; /* The character before p */
        /* Is the list empty? */
        if (charlist->next == 0)
        {
            charlist->next = ch;
            charlist->next->next = 0;
            return;
        }
        /* Possible optimization: Don't start from start of list...
         * start from where at */
        loc = charlist->next;
        /* Is it at the beginning of the list? */
        p = charlist->next;
        /* Load p with the first item on the list */
        if (loc == p->next)
        {
            /* Set the pointers */
            charlist->next = charlist->next;
            charlist->next->next = ch;
            return;
        }
        while (p->next != 0)
        {
            /* We'll find needed point where to place */
            loc = p->next;
            /* Remember last position */
            charlist->next = charlist->next;
            if (p->next == 0)
            {
                /* If it's the end of list, insert it there */
                charlist->next->next = ch;
                charlist->next->next->next = 0;
                return;
            }
            if (p->next == 0)
            {
                /* If member isn't at the end of the list */
                p->next = ch;
            }
        }
    }
}

/* This routine will remove a character from the charlist list
 * loc is not used to place the new character. However,
 * it is used to place the new character. It will
 * place the new character after charlist. After finished, it is an
 * extremely good idea to reset the result!
 */
void CharlistRemoveCharlist(charlist, ch)
{
    struct Charlist *charlist; /* The character */
    struct Charlist *charlist; /* The character */
    int loc;
    struct Charlist *charlist; /* Used for searching for location */
    struct Charlist *charlist; /* The character before p */
    void CharlistRemoveCharlist(charlist, ch)
    {
        struct Charlist *charlist; /* The character */
        struct Charlist *charlist; /* The character before p */
        /* Is the list empty? */
        if (charlist->next == 0)
        {
            return;
        }
        /* Possible optimization: Don't start from start of list...
         * start from where at */
        loc = charlist->next;
        /* Is it at the beginning of the list? */
        p = charlist->next;
        /* Load p with the first item on the list */
        if (loc == p->next)
        {
            /* Set the pointers */
            charlist->next = charlist->next;
            charlist->next->next = 0;
            return;
        }
        while (p->next != 0)
        {
            /* We'll find needed point where to place */
            loc = p->next;
            /* Remember last position */
            charlist->next = charlist->next;
            if (p->next == 0)
            {
                /* If it's the end of list, insert it there */
                charlist->next->next = ch;
                charlist->next->next->next = 0;
                return;
            }
            if (p->next == 0)
            {
                /* If member isn't at the end of the list */
                p->next = ch;
            }
        }
    }
}

/* This routine will calculate the position of a character
 * loc is not used to place the new character. However,
 * it is used to place the new character. It will
 * place the new character after charlist. After finished, it is an
 * extremely good idea to reset the result!
 */
void CharlistCalculateLoc(charlist, ch)
{
    struct Charlist *charlist; /* The character */
    struct Charlist *charlist; /* The character */
    int loc;
    struct Charlist *charlist; /* Used for searching for location */
    struct Charlist *charlist; /* The character before p */
    void CharlistCalculateLoc(charlist, ch)
    {
        struct Charlist *charlist; /* The character */
        struct Charlist *charlist; /* The character before p */
        /* Is the list empty? */
        if (charlist->next == 0)
        {
            return;
        }
        /* Possible optimization: Don't start from start of list...
         * start from where at */
        loc = charlist->next;
        /* Is it at the beginning of the list? */
        p = charlist->next;
        /* Load p with the first item on the list */
        if (loc == p->next)
        {
            /* Set the pointers */
            charlist->next = charlist->next;
            charlist->next->next = 0;
            return;
        }
        while (p->next != 0)
        {
            /* We'll find needed point where to place */
            loc = p->next;
            /* Remember last position */
            charlist->next = charlist->next;
            if (p->next == 0)
            {
                /* If it's the end of list, insert it there */
                charlist->next->next = ch;
                charlist->next->next->next = 0;
                return;
            }
            if (p->next == 0)
            {
                /* If member isn't at the end of the list */
                p->next = ch;
            }
        }
    }
}

```

【例 146】

[illegible]



【図148】

```

// Copy clipped image
for (j=0; j<2; j++)
    temp_image[(1-11)*3-C1] = pixel[(11)*3];
allocate(temp_image, temp_imageC1); // Copy temp image to current image
for (i=0; i<1024; i++)
    for (j=0; j<1024; j++)
        pixel[(11)*3] = temp_image[(11)*3];
}

// Read in part of a TIFF file
void chage::read_tiff(char *filename)
{
    FILE *tagfile; // Input file
    long int tagfile_position; // The our file position (in bytes)
    long int width; // Width of picture
    long int length; // Length of picture
    long int black; // Value of black pixel
    long int hps; // Bytes per strip
    long int start_offset; // Start of image data
    long int nstrips; // Number of strips
    long int type; // Tag type
    if ((tagfile = fopen(filename, "rb")) == NULL)
    {
        cout << "Unable to open file " << filename << "\n";
        exit(1);
    }
    tagfile_position = 0;
    // Read in header and make sure it's local
    if (getbytes(tagfile, tagfile_position, 1) != 0x00000001)
    {
        cout << "... Error: TIFF file not in local format\n";
        exit(1);
    }
    // Read in Magic number
    if (getbytes(tagfile, tagfile_position, 1) != 0x00000001)
    {
        cout << "... Error: TIFF file not in local format\n";
        exit(1);
    }
    // Find first file directory
    if (getbytes(tagfile, tagfile_position, 1) != 0x00000001)
    {
        cout << "... Error: TIFF file not in local format\n";
        exit(1);
    }
    // Tag in Directory
    nstrips = getbytes(tagfile, tagfile_position, 1);
    while (nstrips > 0)
    {
        tag = getbytes(tagfile, tagfile_position, 1);
        type = getbytes(tagfile, tagfile_position, 1);
        value = getbytes(tagfile, tagfile_position, 1);
        if ((tag == 1) && (tag < 32768) == 0)
        {
            cout << "Error: Tag length is 1\n";
        }
    }
}

```

```

// Make the new image (note size adjusted)
allocate(R1, C1, value, 1);
for (int i=0; i<1024; i++)
    pixel[i*3] = value;
}

// Copy a section of a image into another. Image = reference image
// Range is [R1..R2], [C1..C2]
void chage::clip_image(image, int R1, int R2, int C1, int C2)
{
    int r, c;
    if (R2 < R1)
        R2 = R1;
    if (C2 < C1)
        C2 = C1;
    if (C2 > C1)
        C2 = C1;
    if (R2 > R1)
        R2 = R1;
    if (C2 > C1)
        C2 = C1;
    if (R2 > R1)
        R2 = R1;
    if (C2 > C1)
        C2 = C1;
    allocate(R2-R1, C2-C1);
    for (r=0; r<R2-R1; r++)
        for (c=0; c<C2-C1; c++)
            pixel[(r+1)*3] = image_pixel[(r+1)*3 + (c-C1)];
}

// Clipper: reduce the size of an image by removing the black
// border from it. This adjusts the image file.
void chage::clipper()
{
    int R1, R2, C1, C2; // The starting and stopping rows of new image
    int i, j; // Temporary image
    R1 = 0; // Initialise to extremes
    R2 = 0;
    C1 = 0;
    C2 = 0;
    for (i=0; i<1024; i++)
        for (j=0; j<1024; j++)
            if (pixel[i*3] != 0)
            {
                if (i < R1) R1 = i;
                if (i > R2) R2 = i;
                if (j < C1) C1 = j;
                if (j > C2) C2 = j;
            }
    if (R1 > R2) R1 = R2 + 1;
    if (C1 > C2) C1 = C2 + 1;
    temp_allocate(R2-R1, C2-C1); // New image is this size
}

```



【図149】

```

    exit(1);
}
switch (tag)
{
    case 0x00FF:
        if (value != 0)
        {
            cout << "Can't deal with non-standard NewRufFileType\n";
            exit(1);
        }
        break;
    case 0x007F:
        if (value != 1)
        {
            cout << "RufFile type not full resolution\n";
            exit(1);
        }
        break;
    case 0x0100:
        width = value;
        break;
    case 0x0101:
        length = value;
        break;
    case 0x0102:
        if (value != 1)
        {
            cout << "Error: Bits/sample must be 1\n";
            exit(1);
        }
        break;
    case 0x0103:
        if (value != 1)
        {
            cout << "Compression Used Error\n";
            exit(1);
        }
        break;
    case 0x0106:
        black = value;
        break;
    case 0x0107:
        if (value < 1) // (value > 1)
        {
            cout << "Thresholding tag = Literal values\n";
            exit(1);
        }
        break;
    case 0x0111:
        threshold = value;
        break;
    case 0x0117:
        type = value;
        break;
    case 0x0118:
        break;
    case 0x0119:
        break;
    case 0x0121:
        break;
    default:
        if (value & 0xFF00 == 0)
        {
            cout << "!!!!!! Number is unsupported tag appears in the ZIP file\n";
            break;
        }
        // switch
}
} // while

while(tagfile_position < source_offset)
{
    l = getbyte(tagfile, tagfile_position);
}
//----- READ IN FILE -----
allocated(int) length, (int) width;
int v;
for (l = 0; l < length; l++)
{
    for (j = 0; j < width; j++)
    {
        if ((j % 8) == 0)
        {
            v = getbyte(tagfile, tagfile_position);
            if (v == 0xFF)
            {
                cout << "Error: Premature End of File found in this file\n";
                exit(1);
            }
            if ((v & 0x1F) == black_val)
            {
                pixel[l][j] = 0;
            }
            else
            {
                v = v < 1;
            }
        }
    }
    fclose(tagfile);
}
// Read in a byte. File position is updated
// Read in a byte. File position is updated
int change = getbyte(tagfile, long int file_position, int eoferror);
int ci;
ci = getc(f);
if ((ci == EOF) || (eoferror == 1))
{
    cout << "Error: Premature End of File (in getbyte)\n";
    exit(1);
}
// File position++
return ci;
}
// Read in an integer from a file. (LRF, LSR)
long int change = getbyte(FILE, long int ci, long int op);
int ci, ci2;
ci = getbyte(f, op, 1);
ci2 = getbyte(f, op, 1);
return ci * 0x100 + ci2;
}
// Read in a long integer from a file. (LRF, LSR)
long int change = getbyte(FILE, long int ci, long int op);
}

```

【图 167】

【图 151】

-149-

【図 220】

【☒ 2 3 5】



【図 153】

```

for (i=c1->bot - 4; i < c2->bot; i++)
    botpos = c1->curve2[i] + c2->curve2[i];
}

// Compute position and exit
*d3 = botpos - botpos;

if (i==(float)*d3) / ((float) width_small);
if (i < PASC_VALID_DIFF)
    return 1;
else
    return 0;
}

// This routine is called from comb_dlist() with a pair of characters
// that probably isn't an i or j.
// Character. This determines if close enough to be combined.
int comb_dlist_method()
{
    struct Character *c1; // The first character
    struct Character *c2; // The second character
    int d3; // The difference in higher
    int width_small; // The width of the smaller character

    int toppos, botpos; // Positions of the top and bottom
    float f;

    if (d3 == 1)
        // Case when c1 is higher than c2
        // Find bottom pos of c1
        if ((c1->bot - c1->top) < 4)
            botpos = c1->bot;
        else
            botpos = 0;
        for (i=c1->bot - 4; i < c1->bot; i++)
            botpos = c1->curve2[i] + c1->curve2[i];
        // Compute position and exit
        *d3 = toppos - botpos;

        f = ((float)*d3) / ((float) width_small);
        if (i < PASC_VALID_DIFF)
            return 1;
        else
            return 0;

    // Case when c2 is higher than c1
    // Find top pos of c2
    if ((c2->bot - c2->top) < 4)
        toppos = c2->bot;
    else
        toppos = 0;
    for (i=c2->bot - 4; i < c2->bot; i++)
        toppos = c2->curve2[i] + c2->curve2[i];
    // Compute position and exit
    *d3 = toppos - botpos;

    f = ((float)*d3) / ((float) width_small);
    if (i < PASC_VALID_DIFF)
        return 1;
    else
        return 0;

    // Case when c1 is higher than c2
    // Find top pos of c1
    if ((c1->bot - c1->top) < 4)
        toppos = c1->bot;
    else
        toppos = 0;
    for (i=c1->bot - 4; i < c1->bot; i++)
        toppos = c1->curve2[i] + c1->curve2[i];
    // Compute position and exit
    *d3 = toppos - botpos;

    f = ((float)*d3) / ((float) width_small);
    if (i < PASC_VALID_DIFF)
        return 1;
    else
        return 0;

    // Case when c2 is higher than c1
    // Find bottom pos of c2
    if ((c2->bot - c2->top) < 4)
        botpos = c2->bot;
    else
        botpos = 0;
    for (i=c2->bot - 4; i < c2->bot; i++)
        botpos = c2->curve2[i] + c2->curve2[i];
    // Compute position and exit
    *d3 = botpos - toppos;

    f = ((float)*d3) / ((float) width_small);
    if (i < PASC_VALID_DIFF)
        return 1;
    else
        return 0;
}

```





[illegible]

【図157】

```

// Combine 12 and 14, 18181818
if (comb_right_delta (comb.c1, comb.c4, 6(comb.c14)))
{
    comb.c1 = comb.c4, comb.c4 = 6(comb.c14)
    if (comb.c4 == comb.c2)
    {
        // Combine both 12, and 14
        comb_combined12(comb);
        comb_combined14(comb);
        continue;
    }
    // Combine chars 14
    comb_combined14(comb);
    continue;
}
// Combine chars 142
comb_combined12(comb);
continue;
}

////////// Now combine parent signs
if ((comb.c1 == right > comb.c3 == left) && (comb.c2 == right > comb.c1 == left))
{
    // If overlapping
    // Check for parent sign type 1, 14, 18, 1818, 181818
    if (comb_right_overlap(comb.c1, comb.c2)) // Combine 1 and 2
    {
        if (comb.c3 != 0)
        {
            if (comb_right_overlap(comb.c2, comb.c3))
            {
                // Combine 1, 2 and 3
                comb_combined12(comb.c2, comb.c3);
                comb_combined14(comb.c2, comb.c3);
                comb_combined18(comb.c2, comb.c3);
                comb_combined1818(comb.c2, comb.c3);
                comb_combined181818(comb.c2, comb.c3);
                continue;
            }
            comb_combined12(comb);
            continue;
        }
        // Combine c1 & c3
        comb_combined12(comb);
        continue;
    }
    // Check for 0/ part of percent (types 2)
    if (comb_left_percent(comb.c1, comb.c2))
    {
        if (comb.c3 != 0)
        {
            if (comb_right_percent(comb.c2, comb.c3)) // Check for 0/0
            {
                comb_combined12(comb.c2, comb.c3);
                comb_combined14(comb.c2, comb.c3);
                comb_combined18(comb.c2, comb.c3);
                comb_combined1818(comb.c2, comb.c3);
                comb_combined181818(comb.c2, comb.c3);
                continue;
            }
            comb_combined12(comb);
            continue;
        }
        // Just combine 1 & 3
        comb_combined12(comb);
        continue;
    }
    // Check for /0 part of percent
    if (comb_right_percent(comb.c1, comb.c2))
    {
        comb_combined12(comb);
        continue;
    }
    // Just combine 1 & 3
    comb_combined12(comb);
    continue;
}

```

```

////////// Don't combine anything
comb_combined12(comb);
}

```



[illegible]



【図160】

```

// Check for the dash between a character, and break it if needed
void cd_char(
    char* line, // The line
    struct Charlist* clist, // Pointer to charlist
    struct Character* ch, // Pointer to the character we're on (modified
                          // to skip past newly inserted divisions)
    int* hist) // The histogram (previously allocated)
{
    int p;

    // Reject if pixel width is too small.
    if ((ch->right - (ch->left) <= MIN_ABS_WIDTH_TO_CONSIDER)
        return;

#ifdef PRINT_INFO
    ns_out_dbg();
    print_character(line, "ch, 10, 10);
    ns_flush();

    printf("\nCharacter info:\n");
    printf("  top/bot = (%d, %d)\n", (ch->top, (ch->bot);
    printf("  left/right = (%d, %d)\n", (ch->left, (ch->right);
#endif

    // Compute the histogram
    hist_histogram(line, ch, hist);

    // Break left side
    p = cd_left_dash_cut(line, ch, hist);
    if (p != 0)
    {
        cd_cut_character(line, clist, ch, p); // Cut the character
#ifdef PRINT_INFO
        printf("\nLeft dash cut\n");
        printf("\nCharacter info:\n");
        printf("  top/bot = (%d, %d)\n", (ch->top, (ch->bot);
        printf("  left/right = (%d, %d)\n", (ch->left, (ch->right);
#endif
    }

    // Break right side
    p = cd_right_dash_cut(line, ch, hist);
    if (p != 0)
    {
        cd_cut_character(line, clist, ch, p); // Cut the character
#ifdef PRINT_INFO
        printf("\nRight dash cut\n");
#endif
    }
}

//
//.....
// MAIN ROUTINE
//.....
//
// This will cut the dashes of of characters
void cut_dashes(char* line, // The string
    struct Charlist* clist) // The current charlist
{
    int* histogram; // The area of the histogram
    struct Character* charon; // The character we're processing
}

```

【図195】

```

/*
 * Filename: myalloc.h
 * Header file for myalloc.c
 *
 *.....
 */

/* Function definitions for myalloc.c */
#ifdef C99
/* Definition for C++ files */
extern "C" {
    int* ivector(int n);
    void free_ivector(int* v);
    float* fvector(int n);
    void free_fvector(float* v);
    int* ivector_ranged(int n, int a);
    void free_ivector_ranged(int* v, int a);
    int** imatrix(int x, int y);
    void free_imatrix(int** m, int x);
    char** cmatrix(int x, int y);
    void free_cmatrix(char** m, int x);
}
#else
/* Definition for C files */
/* The header file for malloc */
int* ivector();
void free_ivector();
float* fvector();
void free_fvector();
int* ivector_ranged();
void free_ivector_ranged();
int** imatrix();
void free_imatrix();
char** cmatrix();
void free_cmatrix();
#endif

```

【図223】

```

/*.....*/
void serovp_remove(nlist)
struct Member* nlist; /* The list */
{
    free_ivector(nlist->list); /* Deallocate memory */
    nlist->allocated = 0; /* Set so won't be accidentally used */
    nlist->size = 0;
}

/*.....*/
void serovp_add(nlist, a, b)
struct Member* nlist; /* The list */
int a, b; /* Start and stop of the serovp */
{
    if (nlist->size == nlist->allocated) {
        printf("Member* nlist FULL!\n");
        exit(1);
    }
    nlist->list[nlist->size++] = a;
    nlist->list[nlist->size++] = b;
}

```

[図161]

```

loc = loc_count + 1; // for position of the char on the fly)
for (z=0; z<10; z++) {
    while (curvel[z] < curvel[z+1]) // loop until they're equal
        if (line_pixel[z](curvel[z]) > 1) // if there is a non empty space
            break; // then exit (left border not found)
    else
        curvel[z] = 1; // There's an empty space to the right
    while (curvel[z] < curvel[z+1]) // slide the border over to meet it
        if (line_pixel[z](curvel[z]-1) > 1) // Do the same with the right border
            break;
    else
        curvel[z] = 1;
}
if (curvel[z] != curvel[z+1]) // there is a pixel on this line...
{
    loc = curvel[z] + curvel[z+1];
    loc_count++;
}
if (loc_count == 0)
{
    loc = loc + 4 / loc_count; // computes the position (used for .loc;
    pos = loc/4; // compute the rem) position
}
// Find left and right
left = curvel[top];
right = curvel[top];
for (z=top; z<10; z++)
{
    if (curvel[z] < left)
        left = curvel[z];
    if (curvel[z] > right)
        right = curvel[z];
}
if (loc == 0)
{
    loc = 4 * (left+right);
    pos = (left+right)/2;
}
for (z=top; z<10; z++) // fill in the blank lines with pos
    curvel[z] = curvel[z] + pos;
// Copy all the new attributes in
(*ch)-xloc = loc;
(*ch)-xpos = top;
(*ch)-xbot = bot;
(*ch)-xleft = left;
(*ch)-xright = right;
}
// This routine will cut with the character
// and will return the 16 pixels to the right half of the character
// changed line
struct CHARLIST *charl;
// The character list
struct CHARACTER *ch;
// The character to be cut
// char to cut the character
int y;

```

```

// filename: level3.c
// This file contains the routines to do level 3 segmentation (slightly
// modified) using method 1 (Marschner approach)
//
//
// The routines
//
// include: charlist.h
// include: clumps.h
// include: normalloc.h
//
// The constants
#define MAX_CUT_THRESHOLD 2
#define RESPONSE_THRESHOLD 16
//
// This routine will adjust left, right, curvel, curvel, top, bottom
// and pos to represent a new character in the character, then it
// calls charlist_replace to replace the new character
// Possible optimization: usually, this is called when a character is
// void 132, adjust the clipping and side is 'in val'
//
// charlist.h
//
// struct character *ch; // pointer to the character we're on
//
// int top, bot, left, right; // The new left and right stuff.
// int pos; // pos = position, loc = pos + 4
// int loc, loc_count, pos; // Northward, set to (*ch)-curvel and 2
// int flag;
//
// curvel = (*ch)-curvel;
// curvel = (*ch)-curvel;
//
// Adjust the top of the character, if it needs to move down
// flag = 0; // the found when flag == 1,
// for (top = (*ch)-xtop; top < 10; top++)
//     if (curvel[top] < curvel[top+1])
//         if (line_pixel[top](top) > 1) // stop at the first on pixel
//             flag = 1;
//             break;
//
// Adjust the bottom of the character
// flag = 0; // not found when flag == 1,
// for (bot = (*ch)-xbot-1; bot > (*ch)-xbot) if (flag == 0); bot--;
// for (curvel[bot] = 0; curvel[bot] < 0; bot++)
//     if (line_pixel[bot](bot) > 1) // stop at the first on pixel
//         flag = 1;
//         break;
//
// bot++; // Make bot exclusive
//
// Now adjust curvel and curvel, update left and right to 'shrink'
// slightly around the border of the character (move the borders up to the
// // edge of the character.

```

[ 1 6 2 ]

```

struct Character *newchar; // The new character
int i;
newchar = malloc(sizeof(struct Character)); // Set pointers to point to the end
// of the charlist (the next blank character);
for (i=0; i<sizeof(newchar); i++) // Loop through the char (up-down)
{
    newchar->curved[i] = ('ch')->curved[i]; // Break the character
    if (newchar->curved[i] < p) // Make the curves of new char are < p
        newchar->curved[i] = p;
    if (newchar->curved[i] > p) // Make sure curves of old char are < p
        newchar->curved[i] = p;
    if (('ch')->curved[i] > p)
        newchar->curved[i] = p;
}
newchar->top = ('ch')->top;
newchar->bot = ('ch')->bot;
newchar->right = p;
newchar->left = ('ch')->left;
newchar->right = p;
// Adjust ch to the new character
*ch = newchar;

// Compute the histogram of a character
void hist(struct Character *ch)
{
    struct Character **ch;
    int *hist;
    int i, j;
    // Compute the histogram
    for (i=0; i<sizeof(ch); i++)
    {
        hist[i] = 0;
        for (j=0; j<sizeof(ch); j++)
        {
            hist[i] += ch[j];
        }
    }
}

// This will do histogram touching on the characters for level 3 method
void level3(struct Character *ch)
{
    struct Character *ch;
    int *histogram;
    struct Character *ch;
    histogram = malloc(sizeof(int));
    while (ch != 0)
    {
        level3(ch);
        ch = ch->next;
    }
}

// This will do histogram touching on the characters for level 3 method
void level3(struct Character *ch)
{
    struct Character *ch;
    int *histogram;
    struct Character *ch;
    histogram = malloc(sizeof(int));
    while (ch != 0)
    {
        level3(ch);
        ch = ch->next;
    }
}

```

—161—

[illegible]







[illegible]

—164—





[illegible]



【図 1 7 3】

```

break_position = (int)(0) + break_image_size / 2;

// Insert the break
breaklist.add(breaklist, break_position * 2) * vector_height, break_image_size;

// Free memory
free_vector(breaklist);
free_vector(breaklist);
}

// Simple routine used by insert_apex_angle
// Checks a value against the current minimum value. If less, it updates.
// The current minimum
// Where it's at
// It's angle
// The test minimum
// etc...
// Don't substitute if min_at == -1
return;
if (min_at == min)
    min = min;
    min_at = min_at;
    min_theta = min_theta;
}

// The image
// The breaklist
// The apex angle will be forced to
// The apex angle
// The break position (for line computation)
// The location of the break in the breaklist
// Arrays, breaks for break1, and break2
// Allocate memory for a1
// Allocate memory for a2
// Compute break range
// Compute break position
// Ensure that it is within range
if (break1 < 0)
    break1 = 0;
if (break2 < 0)
    break2 = 0;

```

```

// Find vertical projection
findProjection(image, c, minx, startBreak, stopBreak, image.start,
proj_alice, proj, proj_alice, stopProj, c) // Compute vpr

// Find maximum projection (for find peak and valley heights)
max_valley_proj = proj[0]
for (int i=1; i<max_valley_proj; i++)
    max_valley_proj = proj[i]

// Compute max_valley_depth & max_valley_height
compute_max_valley_depth_max_valley_height(max_valley_proj,
max_valley_depth, max_valley_height)

// (max_valley_depth < 0) // Exit if we're too deep in constants
return breakCount;

//==== ENDPOINT ====

//==== FINDING VERTICALS ====
//==== FIND VPR ====
// Now we know a valley exists, so compute angles
find_valley_angles(proj, proj_alice, valley_pos, break, break,
max_valley_pos, max_valley_depth);

// Point at "Position" = valley_pos.proj_alice = 'Angles' = (break_valley_pos)
339 c = (break_valley_pos).valley_pos; // 'Angles' = (break_valley_pos)
c = 339*proj_alice;

// Find minimum of the vpr (break < 0)
// Minimum of the vector (proj, valley_pos, c, proj_alice, min, min(c));
// break = proj_alice;

// Count = vpr minimum at = min(c) = 'min' = 'min'
// Compute the angular vpr from the last break to c, and find minimum
find_min_valley_pos_break(image, break, startBreak-breakCount,
stopBreak-breakCount, c, break, min, min(c));
// If it has a smaller minimum, break, min(c), min, min(c);
break_valley_pos_break(image, break, startBreak-breakCount,
stopBreak-breakCount, c, break, min, min(c));
// Count = c
// If break, min = 'min' = 'min'
// Insert angular comparison into min, min(c), min(c), min(c);
insertAngularComparison(min, min(c), min(c), min(c), break, break);

//==== ENDPOINT ====

//==== CHECK OTHER ANGLES ====
// Double three degrees = 3.14159/180.0 * 3.0;
// Double six degrees = 3.14159/180.0 * 6.0;
// Double nine degrees = 3.14159/180.0 * 9.0;

find_min_valley_pos_break(image, break, startBreak-breakCount,
stopBreak-breakCount, c, break_valley_pos, min, min(c));
// Count = c
// If break, min = 'min' = 'min'
// Insert angular comparison into min, min(c), min(c), min(c);
insertAngularComparison(min, min(c), min(c), min(c), break, break);

//==== CHECK OTHER 2 ====
// Double three degrees = 3.14159/180.0 * 3.0;
// Double six degrees = 3.14159/180.0 * 6.0;
// Double nine degrees = 3.14159/180.0 * 9.0;

find_min_valley_pos_break(image, break, startBreak-breakCount,
stopBreak-breakCount, c, break_valley_pos, min, min(c));
// Count = c
// If break, min = 'min' = 'min'
// Insert angular comparison into min, min(c), min(c), min(c);
insertAngularComparison(min, min(c), min(c), min(c), break, break);

//==== CHECK OTHER 4 ====
// Double three degrees = 3.14159/180.0 * 3.0;
// Double six degrees = 3.14159/180.0 * 6.0;
// Double nine degrees = 3.14159/180.0 * 9.0;

find_min_valley_pos_break(image, break, startBreak-breakCount,
stopBreak-breakCount, c, break_valley_pos, min, min(c));
// Count = c
// If break, min = 'min' = 'min'
// Insert angular comparison into min, min(c), min(c), min(c);
insertAngularComparison(min, min(c), min(c), min(c), break, break);

```

【图 175】

[illegible]





【图 177】

[illegible]

[illegible]



【图 180】

[illegible]



```

//***** PLOTTING ROUTINES *****
void plotChar(
    struct Character *ch, // The image
    int x, // The character to plot
    int y, // The upper left hand corner (left, top) to plot
    int x2,
    int y2,
    int x3,
    int y3,
    int x4,
    int y4,
    int x5,
    int y5,
    int x6,
    int y6,
    int x7,
    int y7,
    int x8,
    int y8,
    int x9,
    int y9,
    int x10,
    int y10,
    int x11,
    int y11,
    int x12,
    int y12,
    int x13,
    int y13,
    int x14,
    int y14,
    int x15,
    int y15,
    int x16,
    int y16,
    int x17,
    int y17,
    int x18,
    int y18,
    int x19,
    int y19,
    int x20,
    int y20,
    int x21,
    int y21,
    int x22,
    int y22,
    int x23,
    int y23,
    int x24,
    int y24,
    int x25,
    int y25,
    int x26,
    int y26,
    int x27,
    int y27,
    int x28,
    int y28,
    int x29,
    int y29,
    int x30,
    int y30,
    int x31,
    int y31,
    int x32,
    int y32,
    int x33,
    int y33,
    int x34,
    int y34,
    int x35,
    int y35,
    int x36,
    int y36,
    int x37,
    int y37,
    int x38,
    int y38,
    int x39,
    int y39,
    int x40,
    int y40,
    int x41,
    int y41,
    int x42,
    int y42,
    int x43,
    int y43,
    int x44,
    int y44,
    int x45,
    int y45,
    int x46,
    int y46,
    int x47,
    int y47,
    int x48,
    int y48,
    int x49,
    int y49,
    int x50,
    int y50,
    int x51,
    int y51,
    int x52,
    int y52,
    int x53,
    int y53,
    int x54,
    int y54,
    int x55,
    int y55,
    int x56,
    int y56,
    int x57,
    int y57,
    int x58,
    int y58,
    int x59,
    int y59,
    int x60,
    int y60,
    int x61,
    int y61,
    int x62,
    int y62,
    int x63,
    int y63,
    int x64,
    int y64,
    int x65,
    int y65,
    int x66,
    int y66,
    int x67,
    int y67,
    int x68,
    int y68,
    int x69,
    int y69,
    int x70,
    int y70,
    int x71,
    int y71,
    int x72,
    int y72,
    int x73,
    int y73,
    int x74,
    int y74,
    int x75,
    int y75,
    int x76,
    int y76,
    int x77,
    int y77,
    int x78,
    int y78,
    int x79,
    int y79,
    int x80,
    int y80,
    int x81,
    int y81,
    int x82,
    int y82,
    int x83,
    int y83,
    int x84,
    int y84,
    int x85,
    int y85,
    int x86,
    int y86,
    int x87,
    int y87,
    int x88,
    int y88,
    int x89,
    int y89,
    int x90,
    int y90,
    int x91,
    int y91,
    int x92,
    int y92,
    int x93,
    int y93,
    int x94,
    int y94,
    int x95,
    int y95,
    int x96,
    int y96,
    int x97,
    int y97,
    int x98,
    int y98,
    int x99,
    int y99,
    int x100,
    int y100,
    int x101,
    int y101,
    int x102,
    int y102,
    int x103,
    int y103,
    int x104,
    int y104,
    int x105,
    int y105,
    int x106,
    int y106,
    int x107,
    int y107,
    int x108,
    int y108,
    int x109,
    int y109,
    int x110,
    int y110,
    int x111,
    int y111,
    int x112,
    int y112,
    int x113,
    int y113,
    int x114,
    int y114,
    int x115,
    int y115,
    int x116,
    int y116,
    int x117,
    int y117,
    int x118,
    int y118,
    int x119,
    int y119,
    int x120,
    int y120,
    int x121,
    int y121,
    int x122,
    int y122,
    int x123,
    int y123,
    int x124,
    int y124,
    int x125,
    int y125,
    int x126,
    int y126,
    int x127,
    int y127,
    int x128,
    int y128,
    int x129,
    int y129,
    int x130,
    int y130,
    int x131,
    int y131,
    int x132,
    int y132,
    int x133,
    int y133,
    int x134,
    int y134,
    int x135,
    int y135,
    int x136,
    int y136,
    int x137,
    int y137,
    int x138,
    int y138,
    int x139,
    int y139,
    int x140,
    int y140,
    int x141,
    int y141,
    int x142,
    int y142,
    int x143,
    int y143,
    int x144,
    int y144,
    int x145,
    int y145,
    int x146,
    int y146,
    int x147,
    int y147,
    int x148,
    int y148,
    int x149,
    int y149,
    int x150,
    int y150,
    int x151,
    int y151,
    int x152,
    int y152,
    int x153,
    int y153,
    int x154,
    int y154,
    int x155,
    int y155,
    int x156,
    int y156,
    int x157,
    int y157,
    int x158,
    int y158,
    int x159,
    int y159,
    int x160,
    int y160,
    int x161,
    int y161,
    int x162,
    int y162,
    int x163,
    int y163,
    int x164,
    int y164,
    int x165,
    int y165,
    int x166,
    int y166,
    int x167,
    int y167,
    int x168,
    int y168,
    int x169,
    int y169,
    int x170,
    int y170,
    int x171,
    int y171,
    int x172,
    int y172,
    int x173,
    int y173,
    int x174,
    int y174,
    int x175,
    int y175,
    int x176,
    int y176,
    int x177,
    int y177,
    int x178,
    int y178,
    int x179,
    int y179,
    int x180,
    int y180,
    int x181,
    int y181,
    int x182,
    int y182,
    int x183,
    int y183,
    int x184,
    int y184,
    int x185,
    int y185,
    int x186,
    int y186,
    int x187,
    int y187,
    int x188,
    int y188,
    int x189,
    int y189,
    int x190,
    int y190,
    int x191,
    int y191,
    int x192,
    int y192,
    int x193,
    int y193,
    int x194,
    int y194,
    int x195,
    int y195,
    int x196,
    int y196,
    int x197,
    int y197,
    int x198,
    int y198,
    int x199,
    int y199,
    int x200,
    int y200,
    int x201,
    int y201,
    int x202,
    int y202,
    int x203,
    int y203,
    int x204,
    int y204,
    int x205,
    int y205,
    int x206,
    int y206,
    int x207,
    int y207,
    int x208,
    int y208,
    int x209,
    int y209,
    int x210,
    int y210,
    int x211,
    int y211,
    int x212,
    int y212,
    int x213,
    int y213,
    int x214,
    int y214,
    int x215,
    int y215,
    int x216,
    int y216,
    int x217,
    int y217,
    int x218,
    int y218,
    int x219,
    int y219,
    int x220,
    int y220,
    int x221,
    int y221,
    int x222,
    int y222,
    int x223,
    int y223,
    int x224,
    int y224,
    int x225,
    int y225,
    int x226,
    int y226,
    int x227,
    int y227,
    int x228,
    int y228,
    int x229,
    int y229,
    int x230,
    int y230,
    int x231,
    int y231,
    int x232,
    int y232,
    int x233,
    int y233,
    int x234,
    int y234,
    int x235,
    int y235,
    int x236,
    int y236,
    int x237,
    int y237,
    int x238,
    int y238,
    int x239,
    int y239,
    int x240,
    int y240,
    int x241,
    int y241,
    int x242,
    int y242,
    int x243,
    int y243,
    int x244,
    int y244,
    int x245,
    int y245,
    int x246,
    int y246,
    int x247,
    int y247,
    int x248,
    int y248,
    int x249,
    int y249,
    int x250,
    int y250,
    int x251,
    int y251,
    int x252,
    int y252,
    int x253,
    int y253,
    int x254,
    int y254,
    int x255,
    int y255,
    int x256,
    int y256,
    int x257,
    int y257,
    int x258,
    int y258,
    int x259,
    int y259,
    int x260,
    int y260,
    int x261,
    int y261,
    int x262,
    int y262,
    int x263,
    int y263,
    int x264,
    int y264,
    int x265,
    int y265,
    int x266,
    int y266,
    int x267,
    int y26
```

```

if (n)
for (i=current_y; i<WINDOW_SIZE_Y; i++)
x=xyloc(current_x-10,i);
for (i=current_x; i<WINDOW_SIZE_X; i++)
for (j=current_y; j<WINDOW_SIZE_Y; j++)
x=xyloc(current_x-11,i);
x=xyloc(current_x-10,i);
current_x = 0; // Maximum of 40 columns over
x=flush();

//===== FIND RECOGNITION ROUTINE =====
// Recognized character defined by from ch to ch1 (inclusive)
// ch1=ch2=ch3=ch4=ch5=ch6=ch7=ch8=ch9=ch10=ch11=ch12=ch13=ch14=ch15=ch16=ch17=ch18=ch19=ch20=ch21=ch22=ch23=ch24=ch25=ch26=ch27=ch28=ch29=ch30=ch31=ch32=ch33=ch34=ch35=ch36=ch37=ch38=ch39=ch40=ch41=ch42=ch43=ch44=ch45=ch46=ch47=ch48=ch49=ch50=ch51=ch52=ch53=ch54=ch55=ch56=ch57=ch58=ch59=ch60=ch61=ch62=ch63=ch64=ch65=ch66=ch67=ch68=ch69=ch70=ch71=ch72=ch73=ch74=ch75=ch76=ch77=ch78=ch79=ch80=ch81=ch82=ch83=ch84=ch85=ch86=ch87=ch88=ch89=ch90=ch91=ch92=ch93=ch94=ch95=ch96=ch97=ch98=ch99=ch100=ch101=ch102=ch103=ch104=ch105=ch106=ch107=ch108=ch109=ch110=ch111=ch112=ch113=ch114=ch115=ch116=ch117=ch118=ch119=ch120=ch121=ch122=ch123=ch124=ch125=ch126=ch127=ch128=ch129=ch130=ch131=ch132=ch133=ch134=ch135=ch136=ch137=ch138=ch139=ch140=ch141=ch142=ch143=ch144=ch145=ch146=ch147=ch148=ch149=ch150=ch151=ch152=ch153=ch154=ch155=ch156=ch157=ch158=ch159=ch160=ch161=ch162=ch163=ch164=ch165=ch166=ch167=ch168=ch169=ch170=ch171=ch172=ch173=ch174=ch175=ch176=ch177=ch178=ch179=ch180=ch181=ch182=ch183=ch184=ch185=ch186=ch187=ch188=ch189=ch190=ch191=ch192=ch193=ch194=ch195=ch196=ch197=ch198=ch199=ch200=ch201=ch202=ch203=ch204=ch205=ch206=ch207=ch208=ch209=ch210=ch211=ch212=ch213=ch214=ch215=ch216=ch217=ch218=ch219=ch220=ch221=ch222=ch223=ch224=ch225=ch226=ch227=ch228=ch229=ch230=ch231=ch232=ch233=ch234=ch235=ch236=ch237=ch238=ch239=ch240=ch241=ch242=ch243=ch244=ch245=ch246=ch247=ch248=ch249=ch250=ch251=ch252=ch253=ch254=ch255=ch256=ch257=ch258=ch259=ch260=ch261=ch262=ch263=ch264=ch265=ch266=ch267=ch268=ch269=ch270=ch271=ch272=ch273=ch274=ch275=ch276=ch277=ch278=ch279=ch280=ch281=ch282=ch283=ch284=ch285=ch286=ch287=ch288=ch289=ch290=ch291=ch292=ch293=ch294=ch295=ch296=ch297=ch298=ch299=ch300=ch301=ch302=ch303=ch304=ch305=ch306=ch307=ch308=ch309=ch310=ch311=ch312=ch313=ch314=ch315=ch316=ch317=ch318=ch319=ch320=ch321=ch322=ch323=ch324=ch325=ch326=ch327=ch328=ch329=ch330=ch331=ch332=ch333=ch334=ch335=ch336=ch337=ch338=ch339=ch340=ch341=ch342=ch343=ch344=ch345=ch346=ch347=ch348=ch349=ch350=ch351=ch352=ch353=ch354=ch355=ch356=ch357=ch358=ch359=ch360=ch361=ch362=ch363=ch364=ch365=ch366=ch367=ch368=ch369=ch370=ch371=ch372=ch373=ch374=ch375=ch376=ch377=ch378=ch379=ch380=ch381=ch382=ch383=ch384=ch385=ch386=ch387=ch388=ch389=ch390=ch391=ch392=ch393=ch394=ch395=ch396=ch397=ch398=ch399=ch400=ch401=ch402=ch403=ch404=ch405=ch406=ch407=ch408=ch409=ch410=ch411=ch412=ch413=ch414=ch415=ch416=ch417=ch418=ch419=ch420=ch421=ch422=ch423=ch424=ch425=ch426=ch427=ch428=ch429=ch430=ch431=ch432=ch433=ch434=ch435=ch436=ch437=ch438=ch439=ch440=ch441=ch442=ch443=ch444=ch445=ch446=ch447=ch448=ch449=ch450=ch451=ch452=ch453=ch454=ch455=ch456=ch457=ch458=ch459=ch460=ch461=ch462=ch463=ch464=ch465=ch466=ch467=ch468=ch469=ch470=ch471=ch472=ch473=ch474=ch475=ch476=ch477=ch478=ch479=ch480=ch481=ch482=ch483=ch484=ch485=ch486=ch487=ch488=ch489=ch490=ch491=ch492=ch493=ch494=ch495=ch496=ch497=ch498=ch499=ch500=ch501=ch502=ch503=ch504=ch505=ch506=ch507=ch508=ch509=ch510=ch511=ch512=ch513=ch514=ch515=ch516=ch517=ch518=ch519=ch520=ch521=ch522=ch523=ch524=ch525=ch526=ch527=ch528=ch529=ch530=ch531=ch532=ch533=ch534=ch535=ch536=ch537=ch538=ch539=ch540=ch541=ch542=ch543=ch544=ch545=ch546=ch547=ch548=ch549=ch550=ch551=ch552=ch553=ch554=ch555=ch556=ch557=ch558=ch559=ch560=ch561=ch562=ch563=ch564=ch565=ch566=ch567=ch568=ch569=ch570=ch571=ch572=ch573=ch574=ch575=ch576=ch577=ch578=ch579=ch580=ch581=ch582=ch583=ch584=ch585=ch586=ch587=ch588=ch589=ch590=ch591=ch592=ch593=ch594=ch595=ch596=ch597=ch598=ch599=ch600=ch601=ch602=ch603=ch604=ch605=ch606=ch607=ch608=ch609=ch610=ch611=ch612=ch613=ch614=ch615=ch616=ch617=ch618=ch619=ch620=ch621=ch622=ch623=ch624=ch625=ch626=ch627=ch628=ch629=ch630=ch631=ch632=ch633=ch634=ch635=ch636=ch637=ch638=ch639=ch640=ch641=ch642=ch643=ch644=ch645=ch646=ch647=ch648=ch649=ch650=ch651=ch652=ch653=ch654=ch655=ch656=ch657=ch658=ch659=ch660=ch661=ch662=ch663=ch664=ch665=ch666=ch667=ch668=ch669=ch670=ch671=ch672=ch673=ch674=ch675=ch676=ch677=ch678=ch679=ch680=ch681=ch682=ch683=ch684=ch685=ch686=ch687=ch688=ch689=ch690=ch691=ch692=ch693=ch694=ch695=ch696=ch697=ch698=ch699=ch700=ch701=ch702=ch703=ch704=ch705=ch706=ch707=ch708=ch709=ch710=ch711=ch712=ch713=ch714=ch715=ch716=ch717=ch718=ch719=ch720=ch721=ch722=ch723=ch724=ch725=ch726=ch727=ch728=ch729=ch730=ch731=ch732=ch733=ch734=ch735=ch736=ch737=ch738=ch739=ch740=ch741=ch742=ch743=ch744=ch745=ch746=ch747=ch748=ch749=ch750=ch751=ch752=ch753=ch754=ch755=ch756=ch757=ch758=ch759=ch760=ch761=ch762=ch763=ch764=ch765=ch766=ch767=ch768=ch769=ch770=ch771=ch772=ch773=ch774=ch775=ch776=ch777=ch778=ch779=ch780=ch781=ch782=ch783=ch784=ch785=ch786=ch787=ch788=ch789=ch790=ch791=ch792=ch793=ch794=ch795=ch796=ch797=ch798=ch799=ch800=ch801=ch802=ch803=ch804=ch805=ch806=ch807=ch808=ch809=ch810=ch811=ch812=ch813=
```













[illegible]

【図190】

```

c->rec = FAILED_RECOGNITION;
ret_value = IL_GOODOUT_39; left(image_list.c.ich); // Do the 99's
return ret_value;
}
if (c == ch)
{
c->rec = FAILED_RECOGNITION; // Incorporate next character to left
c = c->prev;
ret_value--;
}
// Check for exit
if (c == 0) // If we're at the end, exit
return 0;
if (c->next->rec == FAILED_RECOGNITION) // If we've passed a Y
return 0; // Stop searching
if (IL_GOODLONG(c.ch)) // If character is too long
return 0;
}

// This routine works with IL_GOODLONG to check '99's after
// successful recognition. It will detect and remove some errors caused by an false successful recognition.
// and by not checking adjacent characters when can't split at level 5
int IL_GOODLONG_RIGHT(
char* layer,
struct Character *ch,
struct Character *rch)
{
struct Character *c; // The character to start from
int ret_value; // The leftmost character to search (exclusive)
int temp_value; // The 1 of chars removed
ret_value = temp_value = 0;
c = ch;
while(1)
{
c->next; // Get the previous character
temp_value++; // At edge
if (c == rch)
return ret_value;
if (c->next->rec == FAILED_RECOGNITION) // Return if not a M
return ret_value;
if (c->level_out_at != max_thresh_level) // Return if not '99'
return ret_value;
if (IL_GOODLONG(image.c.ch))
{
IL_combine_characters(c.ch); // Combine it into one character (in c)
c->rec = FAILED_RECOGNITION;
ret_value = temp_value;
temp_value = 0;
c = ch;
}
}

// This routine will start at character ch and search to left from ch
// grouping characters to see if they're recognized. If so they are
// combined. Note: ich can == 0.
IL_GOODOUT_39_LEFT(
char* layer,
struct Character *ch,
struct Character *rch)
{
struct Character *c; // The character to start from
int ret_value; // Leftmost character to search (exclusive)
int temp_value; // The leftmost character of the range
ret_value = 0;
c = ch;
while(1)
{
// Start at character we're on
// This loop is 'return'ed out of
if (IL_GOODLONG(image.c.ch)) // c to ch recognized?
{
IL_combine_characters(c.ch); // Combine it into one character (in c)
}
}
}

```





【図192】

```

1
//
//
void level4(
    c_image line,          // The line
    struct Charlist *clist) // The charlist
{
    int proj_min, proj_max, // Used to compute size of proj
    int i;

    if (def_VISIBLE)
        x_clear_vint();
    int w;
    Cursor1_X = Cursor1_Y = 10;
    line_plot(Cursor1_X, Cursor1_Y); // Plot input image
    Cursor2_Y = Cursor1_Y + line.size + 10; // Final output goes at Cursor2
    Cursor2_X = Cursor1_X;
    Cursor2_Y = Cursor2_Y + line.size + 10;
    Cursor3_X = Cursor2_X;
    for (c=0; c<FROM_SIZE_X; c++)
        x_plotxy(c, Cursor2_Y); // Plot border line
    Cursor1_Y = Cursor2_Y + 20;
    Cursor1_Y = Cursor2_Y;
    Cursor1_X = Cursor2_X;
}

if (line.size > MAX_CHAR_HEIGHT)
{
    printf("\n*** ERROR ***\n");
    printf("Line size (%td) is greater than max line size allocated in multilevel\n", line.size);
    printf("**** routines (MAX_CHAR_HEIGHT = %td). Increase MAX_CHAR_HEIGHT, MAX_C\n", MAX_CHAR_HEIGHT);
    exit(3);
}

// Allocate Proj and VPP
i = (NOT_MAX_RANGE/NOT_DEC_RESOLUTION)-1;
proj_min = -dec((1+line.size));
proj_max = line.size+proj_min;
proj = ivector_ranged(proj_min, proj_max);
VPP = ivector(line.size);
VPP_Zero = 1; // Erase the cache

/* ..... Program goes here ..... */
goodout(line, clist);

if (def_VISIBLE)
    // Print final character list
    x_plot_charlist(line, clist, Cursor2_X, Cursor2_Y);
endif
printf("Displaying final level 4 charlist. Enter 0 to continue? ");
// print_m_charlist(clist, clist->fnext); // ***** DELETE ME *****
i = auto_input_int();

// Deallocate
free_vector(VPP);
free_vector_ranged(proj, proj_min);
}

```

【図203】

```

/*
 * The routines in this file determine if a character is recognized or not
 * (for supplementation)
 */
.....

#include "x.h"          /* Include Rich's graphics routines */
#include "autoinput.h"

extern int WINDOW_SIZE_Y;

int recognized(s, size_x, size_c) /* Return a 0 for not recognized, 1 for
    recognized */
{
    int **a;             /* The character matrix a[row][col] */
    int size_x;          /* The number of rows in a */
    int size_c;          /* The number of columns in a */
    int r,c;             /* The row and column of the image */
    int dummy;           /* A dummy variable */

    if (def_VISIBLE)
    {
        /* This routine will print the char at the screen's bottom left corner */
        for (r=0; r<size_x; r++) // Loop through the rows
            for (c=0; c<size_c; c++) // Loop through the cols
                if ((r >= 0) && (r < size_x) && (c >= 0) && (c < size_c))
                    if ((a[r][c] != 0)) // If the pixel is on
                        x_plotxy(c+10, r + WINDOW_SIZE_Y - size_x + 10);
                    else
                        x_unplotxy(c+10, r + WINDOW_SIZE_Y - size_x + 10);
                    else
                        x_unplotxy(c+10, r + WINDOW_SIZE_Y - size_x + 10);

        x_flush(); // Flush the display buffer (to make sure
                    // the character gets printed on the screen *)
    }

    printf("Character displayed. Enter return value (1 = good, 0 = bad)? ");
    dummy = auto_input_int();
    return dummy;
}

```

[ 図 193 ]

```

/* This program contains vector and matrix mallocs */
#include <alloc.h>
#include <stdio.h>
/* Routine for error during memory allocation */
void allocation_error(
    char *s) /* The error text */
{
    int i;
    printf("Memory allocation error %s\n", s);
    fprintf(stderr, "Error: %s\n", s);
    exit(1);
}

/* Allocate an integer vector of size (0..n-1) */
int *vector(int n) /* Size of vector */
{
    int *v;
    v = (int *) malloc(sizeof(int) * n);
    if (!v) allocation_error("In vector()");
    return v;
}

/* Free an integer vector v of size (0..n-1) */
void free_vector(int *v, int n) /* The vector */
{
    free((int *) v);
}

/* Allocate an float vector of size (0..n-1) */
float *vector(float n) /* Size of vector */
{
    float *v;
    v = (float *) malloc(sizeof(float) * n);
    if (!v) allocation_error("In vector()");
    return v;
}

/* Free an float vector v of size (0..n-1) */
void free_vector(float *v, int n) /* The vector */
{
    free((float *) v);
}

/* Allocate an integer matrix of size (0..r-1)(0..c-1) */
int **matrix(int r, int c) /* The size of the matrix */
{
    int **m;
    m = (int **) malloc(sizeof(int *) * r);
    if (!m) allocation_error("In matrix()");
    for (int i = 0; i < r; i++) {
        m[i] = (int *) malloc(sizeof(int) * c);
        if (!m[i]) allocation_error("In matrix()");
    }
    return m;
}

/* Free an integer matrix m of size (0..r-1)(0..c-1) */
void free_matrix(int **m, int r, int c) /* The matrix */
{
    int i;
    for (i = 0; i < r; i++)
        free((int *) m[i]);
    free((int **) m);
}

/* Allocate an char matrix of size (0..r-1)(0..c-1) */
char **matrix(char r, int c) /* The size of the matrix */
{
    char **m;
    m = (char **) malloc(sizeof(char *) * r);
    if (!m) allocation_error("In matrix()");
    for (int i = 0; i < r; i++) {
        m[i] = (char *) malloc(sizeof(char) * c);
        if (!m[i]) allocation_error("In matrix()");
    }
    return m;
}

/* Free an char matrix m of size (0..r-1)(0..c-1) */
void free_matrix(char **m, int r, int c) /* The matrix */
{
    int i;
    for (i = 0; i < r; i++)
        free((char *) m[i]);
    free((char **) m);
}

```

【图 196】

—190—

[图 197]

```

// This routine will return what's in front of where the mouse is facing
// It returns 0 if it is out of range
int t_in_front(char* shape)
{
    int oldc, eldc;
    eldc = t_cursor;
    oldc = t_cursor;
    // Success the old direction
    // What to return
    // Remember this position
    // Move into the new space
    if (t_cursor < 0) { t_cursor = image_size; }
    t_cursor = eldc;
    // Return curve position
    // If too high or low
    return 0;
    if (t_cursor < t_start) {
        t_cursor = t_start;
    }
    // Return curve position
    // It to the left or right of the border
    return 0;
    p = image_size(t_cursor);
    t_cursor = eldc;
    return 0;
}

// Main outlining routine
// Notes: This routine won't outline a pixel if only one is set.
void outline(char* shape)
{
    // Loop of the line
    // The character list
    // The left boundary of the area to outline
    // The right boundary of the area to outline
    // The start of the column of an 'on' pixel
    // On the edge
    struct Character* new_char;
    int top, bot, left, right;
    int curve;
    int start;
    int end;
    // Where the new character is going
    // Used to be put in the new character
    // Points to curve and curve of new char.
    // Points to curve and curve of new char.
    // Where the new character is going
    // Set up complex pointers to the new character we're inserting
    new_char = t_char;
    // Set up complex pointers to the new character we're inserting
    curve = new_char->curve;
    // Set up complex pointers to the new character we're inserting
    curve = new_char->curve;
    // Initialize variables of new outlined character
    for (i=0; i<curve->height; i++) // Initialize curve arrays
    {
        curve[i] = 0;
        curve[i] = 0;
        curve[i] = 0;
    }
    top = bot = 0;
    left = right = 0;
    // Max height
    // Max left and right extremes
    // Initialize turtle stuff
}

```

```

// Initialize curve
// Facing left
// Left boundary
// Right boundary
// Not finished
// While we haven't returned to starting
// Adjust curves for this new position
// Get second bit to 1 (outlined)
// If turtle is left of current break
// Move break
// Update left
// Update top and bot
// (Same for right side)
// Notes: No need to update area
// Side (above) will get it
// Update right
// Check to see if we've entirely circled the large
// If (t_cursor == start) & (t_cursor == end)
// We've returned to the starting point
// Remember direction
// Check the space above
// If (t_in_front(1) & 1) == 1 // If space above is on, and name
// Check space left
// If (t_in_front(1) & 1) == 1 // If something there
// Move to left
// Add loop again
// Check area in front of mouse
// Move forward
// Add loop again
// Check area to right of mouse
// Move to right
// Add loop again

```

## 【図198】

```

turn_right() // Check area to behind of mouse
if ((t_in_front(line) & 5) == 1) // If something there
{
    t_move_forward(); // Move behind mouse
    continue; // And loop again
}
not_finished = 0; // Else it's a one-pixel image. so exit
line.pixel[t_cursor][t_cursor] = line.pixel[t_cursor][t_cursor] + 1;
// Make this spot

// Load the info into the Charlist
new_char->stop = top;
new_char->bot = bot; // (Make it exclusive)
new_char->left = left;
new_char->right = right; // (Make it exclusive)
new_char->type = CHARACTER_PIXEL_OUTLINED;

Charlist_place_new_char(cclist); // Insert the new character

// Mark the character as removed from the list
for (r=0; r<line.size; r++)
for (c=curcol[c]; c < curcol[r]; c++)
    line.pixel[r][c] = line.pixel[r][c] + 1;

}

// =====
// Main routine to call to add breaks to the image
// =====
void do_outlining() // Returns number of characters added
{
    cimage* line; // Image of the line
    struct Charlist* cclist; // The character list
    int start_col; // The range to search through (inclusive)
    int stop_col; // (exclusive)

    int x, c; // The row and column we're on

    for (c=stop_col-1; c>=start_col; c--) // Search through column range
    for (r=line.size-1; r>=0; r--) // Scan from bottom to top
    {
        if (line.pixel[r][c] & 5) == 1) // On pixel that hasn't been outlined
        {
            line.pixel[r][c] = 133;
            print_outlining(line, c);
            extract_outline_char(line, cclist, start_col, stop_col, x, c);
            // Outline the character
        }
    }
}

```

## 【図208】

```

//
// Filename: test_mult.c
// This file is used to test and develop multilevel2.c
// =====
#include <stdio.h>
#include <string.h>
#include <image.h>
#include <ymallev.h>
#include <xs.h>
#include <charlist.h>

extern void l4_initialize();
void level4(cimage* line, struct Charlist* cclist);

void xs_init()
{
    struct Charlist cclist;
    struct Character* ch;
    cimage image;
    int i, j;

    l4_initialize();

    Charlist_initialize(&cclist, 300, 300); // 10 chars, 100-max height

    char filename1[100];
    char filename2[100];
    printf("Input filename? /d3/xif/");
    scanf("%s", filename1);
    strcpy(filename2, "/d3/xif/");
    strcat(filename2, filename1);
    image_read_ifc(filename2);
    image_ellipsepac();

    ch = &(cclist.chr[cclist.size]); // Set new_char to the last character

    // Load the character
    ch->stop = 0;
    ch->bot = image.sizeA;
    ch->left = 0;
    ch->right = image.sizeC;
    for (i=ch->stop; i<ch->bot; i++) {
        ch->curcol[i] = ch->left;
        ch->curcol[i] = ch->right;
    }

    Charlist_place_new_char(&cclist);

    level4(image, &cclist);

    printf("**** End of Program ****\n");
}

```

[illegible]



【201】

```

////////// Step 3a. Find the highest text
a = range(alist); // Using to be high_div
for (i=0; i<DIVISIONS; i++) // If valid cursor
{
    if (cursor[i] < alist[i]) // If it's better
    {
        b = startest(cursor[i]) + stopest(cursor[i]);
        if (b < a)
        {
            a = b;
            top_div = i;
        }
    }
}

for (i=0; i<DIVISIONS; i++) // Initialize include
{
    include[i] = 0; // Don't include anything
    // Include the first one
    include[top_div] = 1;
}

r1 = startest(top_div)(cursor[top_div]); // Initialize the size of the
r2 = stopest(top_div)(cursor[top_div]); // range to this segment
c1 = startest(top_div);
c2 = stopest(top_div);

////////// Step 3b. Loop to left and see what column match
range1 = r1;
range2 = r2;
for (i=top_div-1; i>=0; i--) // Each column to the left
{
    if (cursor[i] < alist[i]) // If we're past the end, don't do it
    {
        range1 = startest(i)(cursor[i]);
        range2 = stopest(i)(cursor[i]);
        if (overlap(range1, range2, range3) > CLOSE_OVERLAP)
        {
            include[i] = 1; // Include this column also
            if (range1 < r1) // Adjust range of r1, r2, c1, c2
            {
                r1 = range1;
                r2 = range2;
            }
            if (range1 > r2)
            {
                r2 = range2;
            }
            if (range1 < c1)
            {
                c1 = startest(i);
            }
            if (range2 > c2)
            {
                c2 = stopest(i);
            }
            range1 = range2;
            range2 = range3;
        }
    }
}

////////// Step 3c. Loop to right and see what column match
range1 = startest(top_div)(cursor[top_div]); // The adjacent division
range2 = stopest(top_div)(cursor[top_div]); // test range.
for (i=top_div+1; i<DIVISIONS; i++) // Each column to the left
{
    if (cursor[i] < alist[i]) // If we're past the end, don't do it
    {
        range1 = startest(i)(cursor[i]);
        range2 = stopest(i)(cursor[i]);
        if (overlap(range1, range2, range3) > CLOSE_OVERLAP)
        {
            include[i] = 1; // Include this column also
            if (range1 < r1) // Adjust range of r1, r2, c1, c2
            {
                r1 = range1;
                r2 = range2;
            }
            if (range1 > r2)
            {
                r2 = range2;
            }
            if (range1 < c1)
            {
                c1 = startest(i);
            }
            if (range2 > c2)
            {
                c2 = stopest(i);
            }
            range1 = range2;
            range2 = range3;
        }
    }
}

```

```

)
////////// Step 3d. Copy the pieces into the image
// Optimize this. Selectively do fill() and clipSpace()
line.fill(0, 21, 21, 255); // Fill up the image with zeros
for (i=0; i<DIVISIONS; i++)
{
    if (include[i] == 1)
    {
        for (r=startcol[i]; r<stopcol[i]; r++)
        {
            line.pixel(r-21)(c-21) = (int) page.pixel(c);
        }
        line.clipSpace(); // Remove white space around the line
    }
}

////////// Step 3e. Do we reject this image?
reject = 0; // Default is accept
if ((r2-r1) < min_line_height)
{
    reject = 1;
    printf("Image range is %d..%d, %d..%d\n", c1, c2, r1, r2);
}

////////// Step 4f. Process line
if (reject == 0)
{
    def VISIBLE
    {
        draw_rect_and_join_clipSpace();
    }
    printf("Enter 0 to run outlining character segmentation, 1 to ignore\n");
    i = auto_input_int();
    if (i == 0)
    {
        process_line(line, client);
    }
    //def OBJ_DRAW
    {
        obj_draw_line(page, line, c1list, c2, c1, c2);
    }
    sendif
    {
        // Skip over all the used sections
        for (i=0; i<DIVISIONS; i++) // Up all the cursors
        {
            if (include[i] == 1) // Skip over the included areas
            {
                continue;
            }
        }
        // Check if we've gone off the end of the list
        end_of_list = 1;
        for (i=0; i<DIVISIONS; i++)
        {
            if (cursor[i] < alist[i]) // If the cursor is not at the end....
            {
                end_of_list = 0;
                // ... don't exit
            }
        }
        draw_vector(include);
        draw_vector(cursor);
        draw_vector(startest, DIVISIONS);
        draw_vector(stopest, DIVISIONS);
        draw_vector(hereas, DIVISIONS);
        draw_vector(stopcol);
        draw_vector(startcol);
    }
}

```



```

MAX_CHAR_LEN = 42;
println("Enter max char len: "); // The maximum character length in pixels
MAX_CHAR_LEN = auto_input_int();

border_ignore = 1;
// c = "input border: width to ignore (in pixels) ";
// cin >> border_ignore;

println("Reading in files... ");
page_readoff_filename();
println("Done.\n");

if (getenv("GWTOUT"))
    print_out(page);

// Initialises page printing routines

// border_ignore = (int) ((float) page_dpi * DT_BORDER_WIDTH / 100);
// println("border ignore = %d\n", border_ignore);

PAGE_A1 = border_ignore; // Define page size
PAGE_A2 = border_ignore;
PAGE_B1 = PAGE_A1;
PAGE_C1 = border_ignore;
PAGE_C2 = page_size * border_ignore;
PAGE_D1 = PAGE_C2 - PAGE_C1;

// ..... THIS IS JUST FOR TESTING .....
let i;
let flush();
println("Enter 0 to continue ");
i = auto_input_int();

// clear_screen_and_plot_image(page);
println("Pausing because page is displayed. Input 0 to segment page ");
border_ignore = auto_input_int();
println();
segment_page(page, actlist);
flush();

println("Program finished. Enter 0 to run again.");
attempt = auto_input_int();
}

Charles.compile(actlist);
auto_compile();

```

—196—





【図206】

```

/* This program will test the charlist */
#include "charlist.h"

void print_charlist(clist)
struct Charlist *clist;
{
    struct Character *ch;

    printf("Charlist:\n");
    ch = clist->first;

    while(ch != 0) {
        printf("Loc: %d, Chars (%-.1) = '%', ch->loc/%i",
            if (ch->prev == 0)
                printf("****");
            else
                printf("%d", ch->prev->stop);
        printf("%d", ch->stop);
        if (ch->next == 0)
            printf("****");
        else
            printf("%d", ch->next->stop);
        printf("\n");
        ch = ch->next;
    }
    printf("End of charlist\n\n");
}

main()
{
    struct Character *ch;
    struct Charlist clist;
    int a(10); /* Dummy vector for height */
    int i,p;

    Charlist_initialize(&clist, 10, 10);

    for (i=0; i<5; i++)
    {
        printf("Position of new char %d: ", i);
        scanf("%d", &p);
        ch = &clist.chr[clist.size];
        ch->stop = i;
        ch->prev = i-1;
        ch->curvel[1] = ch->curvel[1] + p;
        Charlist_place_new_char(&clist);
        print_charlist(&clist);
    }

    while(1)
    {
        printf("Replace which char? ");
        scanf("%d", &i);

        ch = clist.first;
        while(ch != 0) {
            if (ch->stop == i)
                break;
            ch = ch->next;
        }
        if (ch == 0) {
            printf("Character not found.\n");
        } else {
            printf("New position? ");
        }
    }
}

```

【図209】

```

/* This file prints a character to be recognised on the screen by */
/* calling the routine thieu_recognize() */

void thieu_recognize(thieu, size_o, image)
int size_x; /* The number of rows */
int size_y; /* The number of columns */
char **image; /* The image; bit 1 is the image */
{
    int r,c;

    printf("\nThe character is:\n");
    for (r=0; r<size_x; r++)
    {
        for (c=0; c<size_y; c++)
            if (image[r][c] & 1)
                printf(" "); /* On */
            else
                printf("."); /* Off */
        printf("\n");
    }
    printf("***** End of character *****\n\n");
}

```

【图 2 1 4】

```

/* If your routines are in C++ then CPP must be defined in make file */
#ifndef CPP
extern "C" void xs_get_foreground(int x, int y, int b);
extern "C" void xs_get_background(int x, int y, int b);
extern "C" void xs_init();
extern "C" void xs_flush();
extern "C" void xs_gotoxy(int x, int y);
extern "C" void xs_gotoxycol(int x, int y);
extern "C" void xs_redisplay();
extern "C" void xs_clear_screen(int x, int y, int width, int height, char *buff);
extern "C" void xs_clear_win();
extern "C" void xs_clear_box(int x, int y, int width, int height);
extern "C" void xs_attrbg(int x, int y, char *attr);
extern "C" void xs_attrcolor(int color);
extern "C" void xs_win_dim(int *width, int *height);
#endif

```

【図 2 1 0】

[illegible]

—202—

[illegible]

[ 図 2 1 2 ]

```

/* Routine: re_display()
Description: Redisplay the "stuff" that was on the window
Return: Nothing
Author: Rick Lee
*/
re_display()
{
    XClearScreen(display(canvas), XWindow(canvas), 0, 0, 0, TRUE);

    /* Routine: re_draw_box(x, y, width, height)
Description: Draw a box on screen
Return: Nothing
Author: Rick Lee
*/
re_draw_box(x, y, width, height)
{
    int x;
    int y;
    int width;
    int height;
    {
        XDrawRect(display(canvas), XWindow(canvas), line_gc,
            x, y, width, height);
    }

    /* Routine: re_string
Description: Draw a string on the screen
Return: Nothing
Author: Rick Lee
*/
re_string(x, y, str)
{
    char *str;
    XDrawString(display(canvas), XWindow(canvas), line_gc,
        x, y, str, strlen(str));

    /* Routine: re_setcolor
Description: Set the current color
Return: Nothing
Author: Rick Lee
*/
re_setcolor(color)
{
    int color;
    {
        gc.fg = color;
        XSetForeground(display(canvas), line_gc, color);
    }
}

```



【図213】

```

/*
Routine: xs_win_dia
Description: Get the size of the window
Returns: Nothing
Author: Rich Lee
*/
xs_wdia(wid, height)
int *wid;
int *height;
{
    Arg args[2];
    Dimension win_width, win_height;

    XSetArg(args[0], XcWidth, win_width);
    XSetArg(args[1], XcHeight, win_height);
    XGetValues(canvas, args, 2);

    *wid = win_width;
    *height = win_height;
}

/*
xs_set_map[w]
Widget w;
{
    int ncolors;
    XColor colors[256];
    Colormap my_cmap;
    int count = 1;

    Display *d;
    int scr;
    Colormap def_colormap;
    XColor Color;
    int x;
    int the_color = 3000;

    d = XDisplay(w);
    scr = DefaultScreen(d);
    ncolors = DisplayCells(d, scr);
    def_colormap = DefaultColormap(d, scr);
    for(x = 0; x < ncolors; x++)
    {
        colors[x].pixel = x;
        colors[x].flags = DoRed | DoGreen | DoBlue;

        colors[x].red = the_color;
        colors[x].green = the_color;
        colors[x].blue = the_color;
        the_color -= 100;
    }
    XQueryColors(d, def_colormap, colors, ncolors);
    my_cmap = XCreateColormap(d, DefaultRootWindow(d), DefaultVisual(d, scr),
                             AllOfAll);
    XStoreColors(d, my_cmap, colors, ncolors);
}
*/

```

【図228】

```

c1 = get2bytes(f, fp);
c2 = get2bytes(f, fp);
return c1 - c2*65536;
}

// Plot image using Flash Graphics at (xloc, yloc) with color
void cimage::plot(int x, int y)
{
    int px, py;
    char color;

    for (py=0; py<sizeR; py++)
        for (px=0; px<sizeC; px++)
        {
            color = pixel[py][px];
            if (color & 1)
                XDrawPixel(x-px, y-py); // Draw it
        }
}

// Plot image with a box around it (with color 1)
void cimage::plotbox(int x, int y)
{
    int px, py;

    for (px=0; px<sizeC; px++) { // Draw horizontal lines
        if ((x-px-y)>2 && 0)
            XDrawPixel(x-px, y);
        if ((x-px-y)<-2 && 0)
            XDrawPixel(x-px, y+sizeR-1);
    }
    for (py=0; py<sizeR; py++) {
        if ((x-y-py)>2 && 0)
            XDrawPixel(x, y-py);
        if ((x-y-py)<-2 && 0)
            XDrawPixel(x+sizeC-1, y-py);
    }
    plot(x-1, y-1); // Plot the image
}

```

【図240】

```

/* This file prints a character to be recognized on the screen by */
/* calling the routine thicw_recognize() */

char thicw_recognize(size_r, size_c, image)
int size_r; // The number of rows */
int size_c; // The number of columns */
char **image; // The image: bit 1 is the image */
{
    int c;
    char input_string[100];

    printf("\nThe character is:\n");
    for (r=0; r<size_r; r++)
    {
        for (c=0; c<size_c; c++)
        {
            if (image[r][c] & 1)
                printf(" "); // On */
            else
                printf("."); // Off */
            printf("\n");
        }
        printf("***** End of character *****\n\n");
        printf("Enter Character? ");
        scanf("%s", input_string);
        return input_string[strlen(input_string)-1];
    }
}

```



【図216】

```

//
// filename: zerovpp.c
// This routine contains the lines to find blocks that have non-zero vpp
// (level 1 segmentation)
//
// =====
//
#include "charlist.h"
#include "images.h"

// =====
// Scan a line quickly (every 1/4 pixel) looking for an 'on' pixel
int line_scan_quick( // Returns 1 for pixel found, 0 otherwise
    image_t line,      // The line to process
    int c)              // The column to scan
{
    int r;
    for (r=0; r<line.size; r++)
        if ((line.pixel(r)(c) & 1))
            return 1;
    return 0;
}

// =====
// Scan a line completely looking for an 'on' pixel
int line_scan_complete( // Returns 1 for pixel found, 0 for otherwise
    image_t line,        // The line
    int c)               // The column to scan
{
    int r;
    for (r=0; r<line.size; r++)
        if ((line.pixel(r)(c) & 1))
            return 1;
    return 0;
}

// =====
// The main routine. Call this to make the list
void extract_zero_vpp_list(
    image_t line,        // The image of the line to segment
    struct nonzero_vpp_list *list) // A pointer to the list
{
    int c;
    int left, right;
    for (c=0; c<line.size; c++) // Scan the image
    {
        if (line_scan_quick(line, c)) // If found something
        {
            left = right = c; // Look to the left for the start
            while (--c >= 0)
                if (line_scan_complete(line, c) >= 0)
                    break;
            left = c+1;
            c = right; // Keep looking to the right
            while (++c < line.size)
                if (line_scan_complete(line, c) >= 0)
                    break;
            right = c;
            zerovpp_add(list, left, right); // Add in the new break
        }
    }
}

```

【図245】

```

/* If your routines are in C++ then CFI must be defined in make file */
#ifdef CFI
extern "C" void xa_init();
extern "C" void xa_flush();
extern "C" void xa_plotxy(int x, int y);
extern "C" void xa_unplotxy(int x, int y);
extern "C" void xa_redisplay();
extern "C" void xa_display_image(int x, int y, int width, int height, char *buf);
extern "C" void xa_clr_scrn();
extern "C" void xa_grav_box(int x, int y, int width, int height);
extern "C" void xa_string(int x, int y, char *str);
extern "C" void xa_setcolor(int color);
extern "C" void xa_vin_dim(int *width, int *height);
#endif

```

【図218】

```

CC= cc
CFLAGS=
COMPILE_PROFILE = -f-p
COMPILE_STYLE = -g -O
COMPILE_VISIBLE = -DVISIBLE
COMPILE_MAKE_FLAGS = -DMAKE_DATABASE
LDLIBS= -la
LDLIBS2= -lXm -lXt -lX11 -lI -lX -lX11 -lX -lX11 -lX -lX11 -lX -lX11
CFLAGS2= -DCPP $(COMPILE_STYLE) $(COMPILE_VISIBLE) $(COMPILE_PROFILE) $(COMPILE_MAKE_FLAGS)
OBJECTS_courcut.o pssegment.o lineprocess.o zerovpp.o charlist.o images.o mymalloc.o
xs.o xs.o util.o autolinput.o courcut.o thieu_file.o

all: courcut

courcut: $(OBJECTS_courcut)
$(CXX) -o courcut $(OBJECTS_courcut) $(LDLIBS) $(LDLIBS2)

autolinput.o: autolinput.c
$(CC) -c $(CFLAGS) autolinput.c

images.o: images.c
$(CXX) -c $(CXXFLAGS) images.c

courcut.o: courcut.c
$(CXX) -c $(CXXFLAGS) courcut.c

lineprocess.o: lineprocess.c
$(CXX) -c $(CXXFLAGS) lineprocess.c

mymalloc.o: mymalloc.c
$(CC) -c $(CFLAGS) mymalloc.c

pssegment.o: pssegment.c
$(CXX) -c $(CXXFLAGS) pssegment.c

scaleprint.o: scaleprint.c
$(CXX) -c $(CXXFLAGS) scaleprint.c

thieu_file.o: thieu_file.c
$(CC) -c $(CFLAGS) thieu_file.c

util.o: util.c
$(CC) -c $(CFLAGS) util.c

xs.o: xs.c
$(CC) -c $(CFLAGS) xs.c

xs.o: xs.c
$(CC) -c -DCPP $(CFLAGS) xs.c

zerovpp.o: zerovpp.c
$(CXX) -c $(CXXFLAGS) zerovpp.c

charlist.o: charlist.h mymalloc.h
combine.o: charlist.h images.h
images.o: images.h xs.h
level.o: mymalloc.h
lineprocess.o: charlist.h images.h
multilevel.o: charlist.h xs.h mymalloc.h
multilevel.o: charlist.h images.h xs.h mymalloc.h
outline.o: charlist.h images.h
pssegment.o: images.h xs.h mymalloc.h
scaleprint.o: charlist.h images.h
zerovpp.o: charlist.h images.h

```

[図219]

```

/* filename: autoinput.h
 * This file contains the code to input from a filename
 * .....
 */
#include <stdio.h>
#include "autoinput.h"
/* Global variables */
int auto_eof_detected;
FILE *auto_fp;
int save_to_file;
int save_count;
char filename[100];
/* .....
void auto_initialize()
{
    int not_exit = 1;
    auto_eof_detected = 0;
    while(not_exit)
    {
        printf("Input filename for autoinput? ");
        scanf("%s", filename);
        auto_fp = fopen(filename, "r");
        if (auto_fp == NULL) /* lead open failed */
        {
            auto_fp = fopen(filename, "w");
            if (auto_fp == NULL) /* write open failed */
            {
                printf("cannot open file %s\n", filename);
                printf("Please check the filename and file\n");
                printf("number 1 at the next prompt.\n");
                save_to_file = 1;
                auto_eof_detected = 1;
                not_exit = 0;
            }
            else {
                printf("opening existing file %s\n", filename);
                not_exit = 0;
            }
        }
    }
}

printf("Do you wish to save the auto-input into this file? (1 = yes) ");
save_to_file = auto_input_int();
printf("Warning: Change the lit '1' to a '0' to turn off appending to the file.\n");
save_count = 0;
}

int auto_input_int()
{
    int i;
    if (auto_eof_detected)
    {
        printf("\nWarning: Input (-9 = newline, -9999 = exit) ? ");

```

```

scanf("%d", &i);
if (i == -9999)
{
    fclose(auto_fp);
    exit(0);
}
if (save_to_file == 1)
{
    if (i == -9) /* Print new lines */
    {
        printf(auto_fp, "\n"); /* Print the new lines */
        fflush(auto_fp);
        save_count = 0;
        return auto_input_int(); /* Add input again */
    }
    printf(auto_fp, "%d ", i);
    fflush(auto_fp);
    if (++save_count > 10)
    {
        save_count = 0;
        printf(auto_fp, "\n");
        fflush(auto_fp);
        return 1;
    }
}
if (fclose(auto_fp, "rd", &i) == EOF)
{
    printf("\n----- DO OF FILE DETECTED -----");
    if (save_to_file == 1)
    {
        fclose(auto_fp);
        auto_fp = fopen(filename, "a");
        if (auto_fp == NULL)
        {
            printf("----- WARNING: CANNOT APPEND TO FILE, IS ...");
            return 1;
        }
        printf(auto_fp, "\n");
        save_count = 0;
        return auto_input_int();
    }
    printf("\n");
    return 1;
}

```

[ 2 2 1 ]

```

/* filename: charlist.c
 * This file contains CharList and CharListUtility utility routines
 * .....
 */
/* .....
 * Routines to handle Character Lines
 * .....
 */
/* By Christopher Smithey */
#include <math.h>
#include <malloc.h>

/* .....
 * Initialize and allocate a CharList (set up curve pointers) */
/* Call this routine once only! */
void CharList_initialize(CharList *clist, CharList *vector_height)
{
    struct CharList *clist; /* The CharList to be initialized */
    int i; /* Looping var */
    int vector_height; /* Height of the line (to allocate curves) */

    /* .....
     * A looping var */
    for (i = 0; i < vector_height; i++)
    {
        /* Allocate memory for characters */
        clist[i] = (struct CharList *) malloc(sizeof(struct CharList));
        if (!clist[i])
        {
            printf("Memory Allocation Error in CharList_initialize()\n");
            exit(1);
        }
        clist[i]-size = 0; /* Number of characters */
        clist[i]-allocated = clist[i]-size; /* Size allocated */
        clist[i]-first = 0; /* No first character */
        clist[i]-vector_height = vector_height;

        /* Allocate memory for curves and set pointers */
        /* NOTE: To optimize, hardcode the array into the code */
        clist[i]-curve = (vector_height * 4) * (vector_height);
        clist[i]-curve = (vector_height * 4) * (vector_height);
    }
}

/* .....
 * Free previously allocated CharList */
void CharList_free(CharList *clist)
{
    struct CharList *clist;
    clist->size = 0; /* Number of characters */
    clist->first = 0; /* No first character */
}

/* .....
 * Terminate and free the CharList */
void CharList_terminate(CharList *clist)
{
    struct CharList *clist;
    int i;

    for (i = 0; i < clist->size; i++)
    {
        free_jvector(clist->curve[i].curve);
        free_jvector(clist->curve[i].curve);
    }
}

```

```

/* .....
 * Free CharList */
void CharList_free(CharList *clist)
{
    struct CharList *clist;
    int i;

    /* Place a new character in the CharList */
    /* Takes the character beyond the CharList (charlist)
     * and inserts it into the list */
    void CharList_place_new_char(CharList *clist, char ch)
    {
        struct CharList *clist; /* Pointer to the new character adding */
        newchar = *clist->charlist->newchar; /* Set newchar */
        /* Calculate the position of the new character */
        CharList_recalculate_loc(newchar);

        /* Check for full list */
        /* Increment the size of the list */
        (clist->size)++;
        /* If list-size is clist->allocated, if too big, */
        printf("Error in CharList_place_new_char(): Character list is FULL!\n");
        exit(1);
    }

    /* Place new in the list character */
    CharList_place_new_char(clist, newchar);

    /* .....
     * This routine repositions a character in the CharList. It removes
     * the character, recalculates loc and inserts it in the list accordingly
     * void CharList_replace_char(CharList *clist, char ch)
     * struct CharList *clist;
     * struct CharList *ch;

    /* Recalculate the position of the character */
    CharList_recalculate_loc(ch);
    if (clist->size < 1) /* If it's the only item in the list */
        return;

    /* Remove the character from the list */
    if (ch->prev == 0) /* If it's on list */
        clist->first = clist->first->next;
    else
    {
        clist->first->prev = ch->prev;
        ch->prev->next = ch->next;
        if (ch->next == 0)
            clist->last = ch->prev;
    }

    /* Place the character in the list */
    CharList_place_new_char(clist, ch);
}

/* .....
 * Insert a character after doing a cut */

```

【図222】

```

/* This routine will insert the character beyond the charlist (i.e. charlist).
 * loc is calculated for both character and the new character; however,
 * loc is NOT USED TO PLACE THE CHARACTER. It will
 * place the new character after character. After finished, it is an
 * extremely good idea to reset the breaklist
 */
void Charlist_Insert_After_Out(Charlist, charater)
{
    struct Charlist *ch; /* Place the new character after this one */
    struct Charlist *newchar; /* Pointer to the new character adding */
    newchar = (struct Charlist *) malloc(sizeof(struct Charlist)); /* Set newchar */
    /* Calculate the position of the new character */
    Charlist_Calculate_Loc(newchar);
    Charlist_Calculate_Loc(charater);
    /* Check for full list */
    if (charater == '\0') /* Increment the size of the list */
    {
        print("Error in Charlist_place_new_char(): Character list is FULL!\n");
        exit(1);
    }
    newchar->prev = charater;
    newchar->next = charater->next;
    charater->next = newchar;
    if (newchar->next != 0)
        newchar->next->prev = newchar;
}

/* This routine will remove a character from the Charlist.
 * Note that only the character is removed, not the pointers.
 * Also, prev and next pointers of char are NOT altered!
 */
void Charlist_Remove_Charlist(char)
{
    struct Charlist *ch;
    struct Charlist *ch2;
    if (ch->prev == 0) /* If list on list */
        ch->first = ch->first->next;
    else
        ch->first->prev = ch;
    if (ch->next != 0)
        ch->next->prev = ch->prev;
}

/* This routine will recalculate the position of a character */
void Charlist_Rcalculate_Loc(char) /* The character */
{
    struct Charlist *ch;
    int i, loc;
    /* Calculate the position of the new character */
    for (i=0; i<ch->size; i++) /* Position = average position.. */
        loc = ch->prev[i] + ch->next[i]; /* .. of the curve */
    loc = loc / 2 / (ch->next - ch->prev); /* Set it */
    ch->loc = loc;
}

```

```

/* This routine will place a character (that is not in the charlist)
 * beyond the charlist.
 * struct Charlist *ch; /* The character */
int loc; /* Used for searching for location */
struct Charlist *last_p; /* The character before p */
/* Is the list empty? */
if (ch->size == 1)
{
    ch->next = 0;
    ch->first = ch;
    return;
}
/* Possible optimization, don't start from start of list...
 * last_p = ch->loc; /* Retrieve position */
/* Is it at the beginning of the list? */
p = ch->first;
/* Load p with the first item on the list */
/* If it's the first on the list */
{
    p->prev = 0;
    ch->next = ch->first;
    ch->prev = 0;
    ch->first = ch;
    return;
}
/* Set the pointers */
/* Set the loc on the list to this */
/* Until find record past where to place */
while (p->loc < loc)
{
    last_p = p;
    p = p->next;
    /* Get the next character */
    /* If it's the end of list, insert it there */
    break;
}
ch->prev = last_p;
ch->next = p->next;
last_p->next = ch;
if (p != 0)
    p->prev = ch;
}

/* Allocate memory for list */
size = size; /* Compute memory allocated, etc */
newlist = (struct Charlist *) malloc(sizeof(struct Charlist)); /* Allocate memory */
newlist->first = newlist->next; /* Set next allocated */
newlist->prev = 0; /* Set prev allocated */
newlist->size = 0; /* Set to zero elements */
int size; /* The number of entries (is allocated) */

```

[illegible]





【図 2 2 6】

```

// MAIN the new image (new size adjusted)
for (int i=0; i<newC1; i++)
    for (int j=0; j<newC2; j++)
        pixel[i][j] = value;

//
//
//
// Copy a section of a image into another. image = reference image
// Range is [R1..R2] [C1..C2]
void image::clip(image* image, int R1, int R2, int C1, int C2)
{
    int R,C,C1;

    if (R2 < R1)
        // Swap R1 & R2
    if (C2 < C1)
        // Swap C1 & C2
    if (R1 < 0) R1 = 0;
    if (R2 > R) R2 = R;
    if (C1 < 0) C1 = 0;
    if (C2 > C) C2 = C;
    if (R1 > image->R) R1 = image->R;
    if (R2 > image->R) R2 = image->R;
    if (C1 > image->C) C1 = image->C;
    if (C2 > image->C) C2 = image->C;

    allocate(R2-R1, C2-C1);

    for (int i=R1; i<R2; i++)
        for (int j=C1; j<C2; j++)
            pixel[i][j] = image->pixel[i-R1][j-C1];
}

// ClipSpace, reduce the size of an image by removing the blank
// space around it. This adjusts the image size.
void image::clipSpace()
{
    int R1, R2, C1, C2; // The starting and stepping row of new image
    int i, j;
    struct image temp; // Temporary image

    R1 = 0;
    R2 = image->R;
    C1 = 0;
    C2 = 0;

    for (i=0; i<image->R; i++)
        for (j=0; j<image->C; j++)
            if (pixel[i][j] != 0)
            {
                if (i < R1) R1 = i;
                if (i > R2) R2 = i;
                if (j < C1) C1 = j;
                if (j > C2) C2 = j;
            }

    if (R1 > R2) R1 = R2 = 0;
    if (C1 > C2) C1 = C2 = 0;

    temp.allocate(R2-R1, C2-C1); // New image is this size
    for (i=R1; i<R2; i++)

```



【図229】

```

// Filename: cimages.h
// This is the header file for cimages.C
//
//
// =====
// #define MDX_DEBUG 1
#include <stream.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

class cimage; // Forward reference

//----- CLASS: cimage -----
//----- CLASS: cimage -----
class cimage
{
public:
    int      sizeR; // Number of rows (first index of array)
    int      sizeC; // Number of cols. (second index of array)
    int      allocated_sizeR; // Amount allocated
    int      allocated_sizeC; // Amount allocated
    int      dpi; // Dots per inch of the image (0 if unknown)
    unsigned char **pixels; // The binary image (2d array)

    cimage(); // Initialize
    ~cimage(); // Destroy
    cimage(int r, int c); // Initialize with a size
    cimage(cimage& temp);

    void reallocate(int R, int C); // Will allocate memory for an image
    void fill(char value, int R, int C); // Fill image with a constant value
    void clip(cimage& image, int R1, int R2, int C1, int C2);
    void clipmask(); // Clips zero space surrounding image
    void readtiff(char *filename); // Read in a tiff image
    void plot(int x, int y); // Plot binary image
    void plotbox(int x, int y); // Plot with a box around it

private:
    void allocate(int r, int c); // Allocate memory
    void deallocate(); // Deallocate memory
    int getbyte(FILE *f, long int& file_position, int &error);
    long int getbytes(FILE *f, long int& fpi);
    long int getbytes(FILE *f, long int& fpi);
};

```











[illegible]





【図239】

```

int border_ignore;
int i;

charlist_initialize(&elist, MAX_CHARS_PER_LINE, MAX_LINE_HEIGHT);
Process_line.fill(0,255); // Allocate the memory for the lines

xs_win_dim(WINDOW_SIZE_X, &WINDOW_SIZE_Y); // Get the window size

printf("Input tiff filename? /d:/tiff/");
strcpy(filename1, "/d:/tiff/twsh1.tif");
scanf("%s", filename1);
strcpy(filename, "/d:/tiff/");
strcpy(filename, filename1);

auto_initialize(); // Initialize Auto-input routines

printf("Enter court average width? ");
COURT_AVG_WIDTH = auto_input_int();

printf("Enter court delta? ");
COURT_DELTA = auto_input_int();

page.dpi = 100;

border_ignore = 1;
// cout << "Input border width to ignore (in pixels)? ";
// cin >> border_ignore;

printf("Reading in File... ");
page.readtiff(filename);
printf("Done.\n");

// segm_init(page); // Initialize page printing routines

// border_ignore = (int) ((float) page.dpi * DEF_BORDER_IGNORE);
// printf("border ignore = %d\n", border_ignore);

PAGE_X1 = border_ignore; // Define page size
PAGE_X2 = page.sizeX - border_ignore;
PAGE_X3 = PAGE_X2 - PAGE_X1;
PAGE_C1 = border_ignore;
PAGE_C2 = page.sizeC - border_ignore;
PAGE_C3 = PAGE_C2 - PAGE_C1;

// ***** THIS IS JUST FOR TESTING *****
xs_flush();
printf("Enter 0 to continue? ");
i = auto_input_int();

// clear_scale_and_plot_image(page);

printf("Pausing because page is displayed. Input 0 to segment page? ");
border_ignore = auto_input_int();

while(1) {
    segment_page(page, &elist);
    xs_flush();

    printf("Program finished. Enter 0 to run again? ");
    int ttttemp;
    ttttemp = auto_input_int();
}

charlist_terminate(&elist);
auto_terminate();
}

```

【図241】

```

/* The character to place bits from */
int p;
/* The position of the bit number */
int n;
return (k >> (p-1)) & -1 <= 0;
}

/* Routine: ut_ascii_bitmap(width,height,buf)
Description: Display the bitmap using ASCII characters
Return: Nothing
Author: Rick Lee
*/
ut_ascii_bitmap(width,height,buf)
int width;
int height;
char *buf;
{
    short bit_code = 0x0000;
    short bit_result;
    short n;
    for (n = 0; n < height; n++)
    {
        for (x = 0; x < width; x++)
        {
            bit_code = bit_code << 1;
            bit_result = bit_code & 0x0000;
            if (ut_getbit(x,n))
                bit_result |= 1;
            buf[n*width+x] = bit_result;
        }
    }
}

/* Routine: ut_reverse(width,height,buf)
Description: Reverse the bits for display format
Return: Nothing
Author: Rick Lee
*/
ut_reverse(width,height,buf)
int width;
int height;
char *buf;
{
    unsigned char val,new_val;
    int x,y;
    for (x = 0; x < (width-1); x++)
    {
        for (y = 0; y < height; y++)
        {
            val = buf[x];
            new_val = 0;
            for (z = 0; z < 8; z++)
            {
                if (ut_getbit(x,y))
                    new_val |= 1 << (7-z);
            }
            buf[x] = new_val;
        }
    }
}

/* Routine: ut_getbit(x,n)
Description: Check if the bit is turned on or not
Return: Nothing
Author: Rick Lee
*/
ut_getbit(x,n)

```

—225—

—225—

【図243】

```

XCopyArea(XtDisplay(canvas), pix, XtWindow(canvas), line_gc, 0,
width, height, x, y);

/* Now free the bitmap */
XFreePixmap(XtDisplay(canvas), bitmap);

/* Now free the bitmap */
XFreePixmap(XtDisplay(canvas), pix);

/* Now flush the queue to X-server */
XFlush(XtDisplay(canvas));
}

/*
Routine: xs_clr_win()
Description: Clear the screen
Return: Nothing
Author: Rick Lee
*/
xs_clr_win()
{
XClearArea(XtDisplay(canvas), XtWindow(canvas), 0, 0, 0, TRUE);
}

/*
Routine: xs_draw_box(x, y, width, height)
Description: Draw a box on screen
Return: Nothing
Author: Rick Lee
*/
xs_draw_box(x, y, width, height)
int x;
int y;
int width;
int height;
{
XDrawRectangle(XtDisplay(canvas), XtWindow(canvas), line_gc,
x, y, width, height);
}

/*
Routine: xs_string
Description: Draw a string on the screen
Return: Nothing
Author: Rick Lee
*/
xs_string(x, y, str)
int x;
int y;
char *str;
{
XDrawString(XtDisplay(canvas), XtWindow(canvas), line_gc,
x, y, str, strlen(str));
}

/*
Routine: xs_setcolor
Description: Set the current color
Return: Nothing
Author: Rick Lee
*/
xs_setcolor(color)
int color;
{
gcv.foreground = color;
XSetForeground(XtDisplay(canvas), line_gc, color);
}

/*
Routine: xs_win_dim
Description: Get the size of the window
Return: Nothing
Author: Rick Lee
*/
xs_win_dim(width, height)
int *width;
int *height;
{
Arg args[2];
Dimension win_width, win_height;
XSetArg(args[0], XtNwidth, win_width);
XSetArg(args[1], XtNheight, win_height);
XtGetValues(canvas, args, 2);

*width = win_width;
*height = win_height;
}

/*
xs_set_map(u)
Widget w;
{
int ncolors;
XColor colors[256];
Colormap my_map;
int count = 1;

Display *d;
int scr;
Colormap def_colormap;
XColor Color;
int w;
int the_color = 2010;

d = XtDisplay(w);
scr = DefaultScreen(d);
ncolors = DisplayCells(d, scr);
def_colormap = DefaultColormap(d, scr);
for(x = 0; x < ncolors; x++)
{
colors[x].red = ...
}
}

```

[illegible]



【図247】

```

//
// File: lzwvpp.c
// This routine contains the lines to find blocks that have non-zero vpp
// (line) segmentation
//
// =====
//
#include "charlist.h"
#include "cinaps.h"

// =====
// Scan a line quickly (every 1/4 pixel) looking for an 'on' pixel
int line_scan_quick( // Returns 1 for pixel found, 0 otherwise
cinaps line, // The line to process
int c) // The column to scan
{
    int r;
    for (r=0; r<line.size; r++)
        if (line.pixel(r)(c) & 1)
            return 1;
    return 0;
}

// =====
// Scan a line completely looking for an 'on' pixel
int line_scan_complete( // Returns 1 for pixel found, 0 for otherwise
cinaps line, // The line
int c) // The column to scan
{
    int r;
    for (r=0; r<line.size; r++)
        if (line.pixel(r)(c) & 1)
            return 1;
    return 0;
}

// =====
// The main routine. Call this to make the list
void extract_zero_vpp_list(
cinaps line, // The image of the line to segment
struct vpplist *list) // A pointer to the list
{
    int c;
    int left, right;
    for (c=0; c<line.size; c++) // Scan the image
    {
        if (line_scan_quick(line, c) // If found something
        {
            left = right = c; // Look to the left for the start
            while (--c >= 0)
                if (line_scan_complete(line, c) == 0)
                    break;
            left = c;
            c = right; // Keep looking to the right
            while (++c < line.size)
                if (line_scan_complete(line, c) == 0)
                    break;
            right = c;
            vpp_add(list, left, right); // Add in the new break
        }
    }
}

```

【手続補正書】

【提出日】平成5年5月13日

【手続補正1】

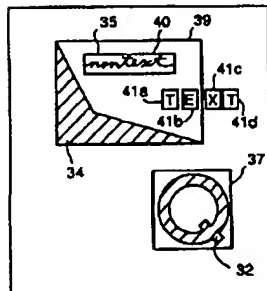
【補正対象書類名】図面

【補正対象項目名】全図

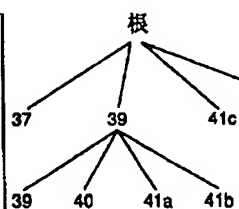
【補正方法】変更

【補正内容】

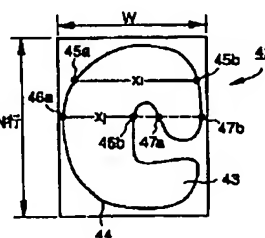
【図5B】



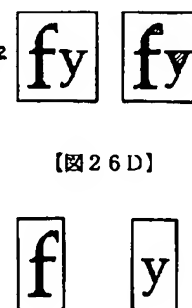
【図5C】



【図6A】



【図26B】【図26C】



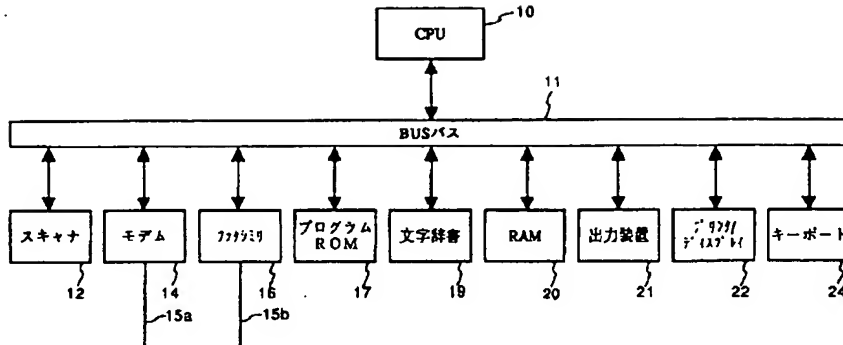
【図26D】



【図1】 \* \*

【図35】

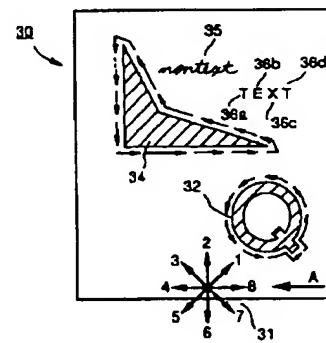
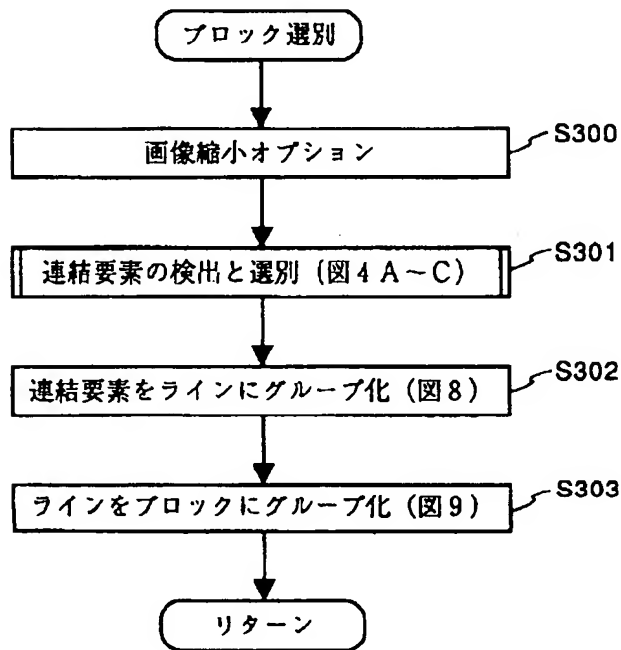
Source Code For  
Block Selection



【図3】

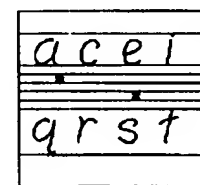
※ ※

【図5A】



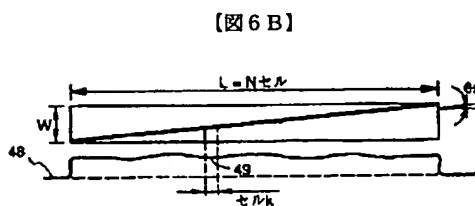
【図18B】

【図234】



【図6C】

【図194】



【図6B】

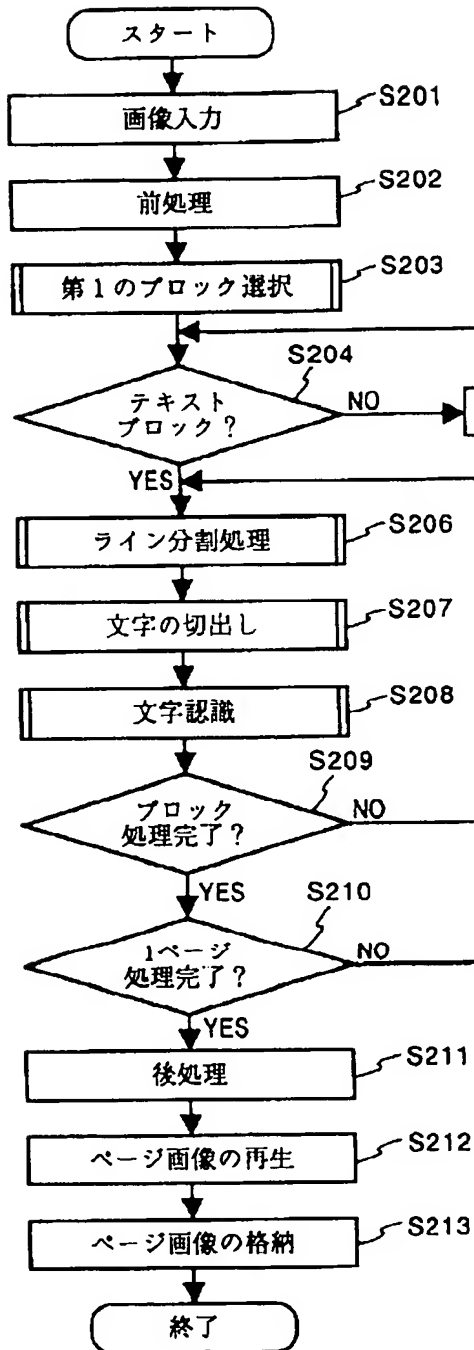
【図126】

```

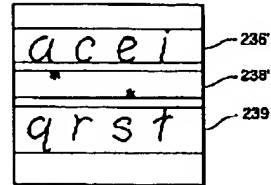
#define FILE_LENGTH      80
#define LINK_LENGTH      256
#define FONT_SIZE_LENGTH 30

```

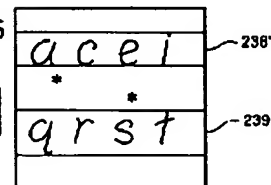
【図2】



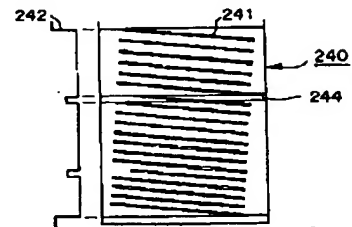
【図18C】



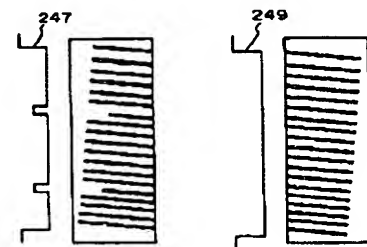
【図18D】



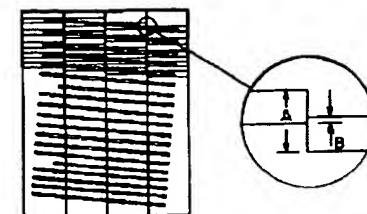
【図19A】



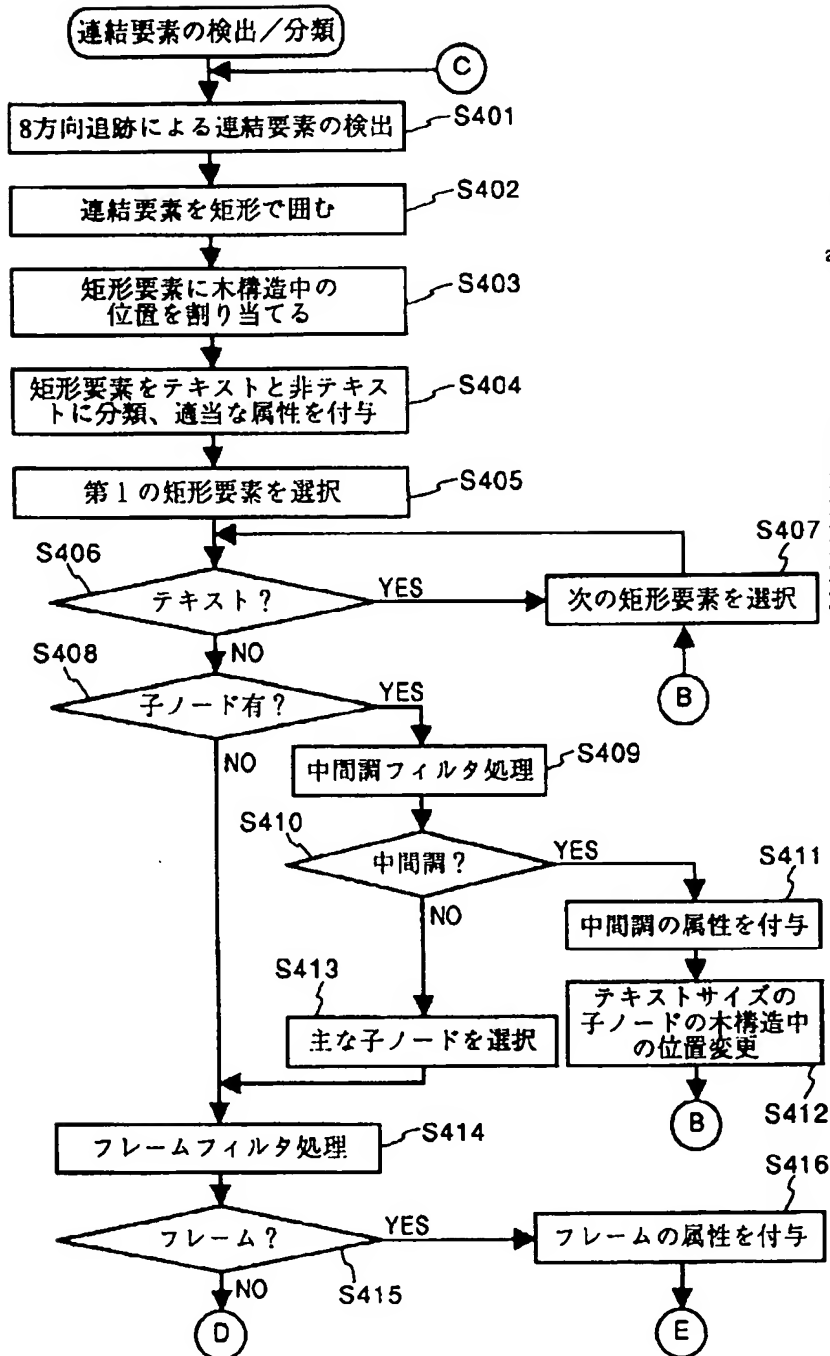
【図19B】



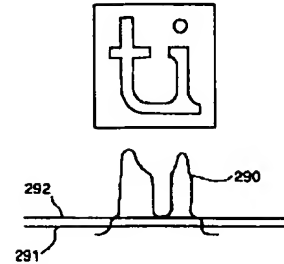
【図19D】



【図4A】



【図29A】

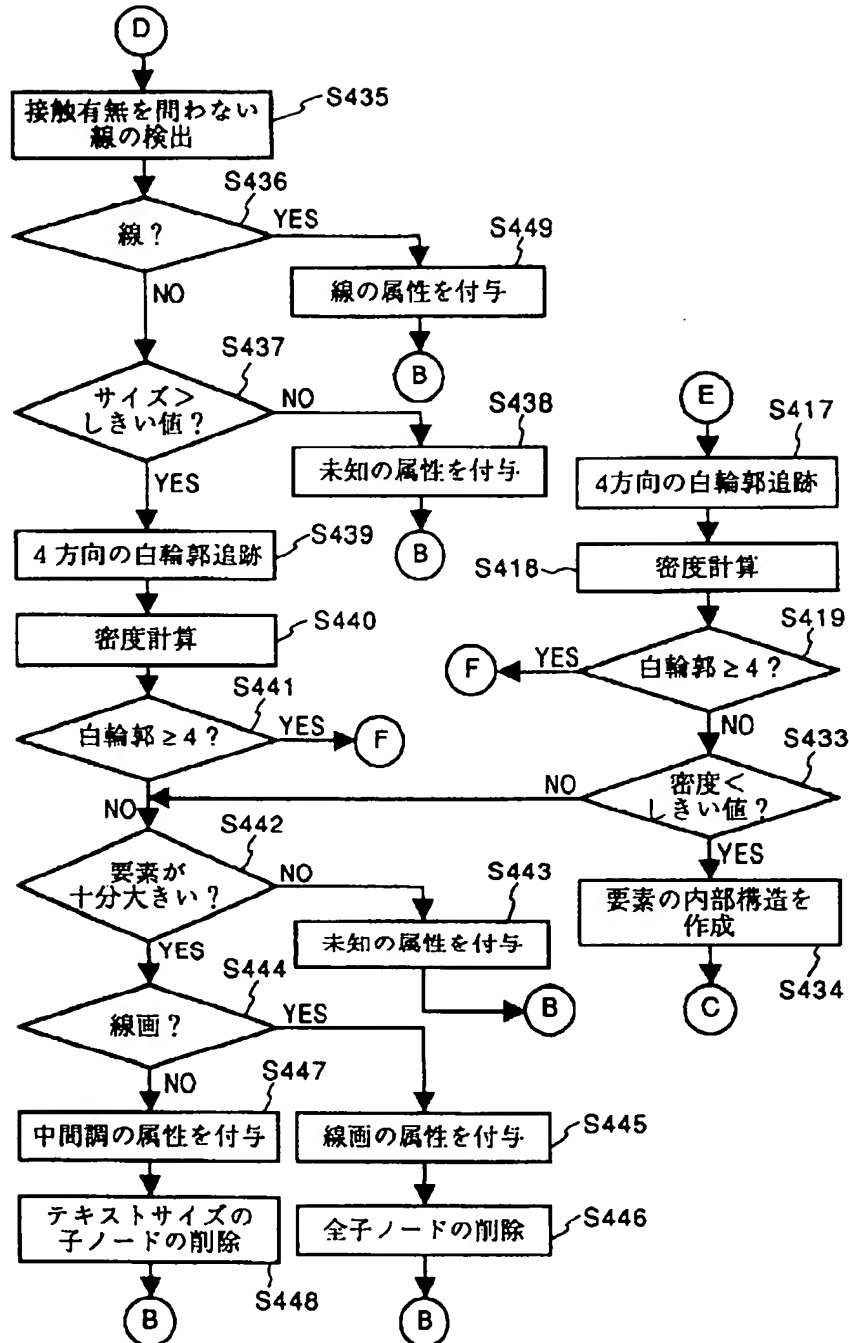


【図134】

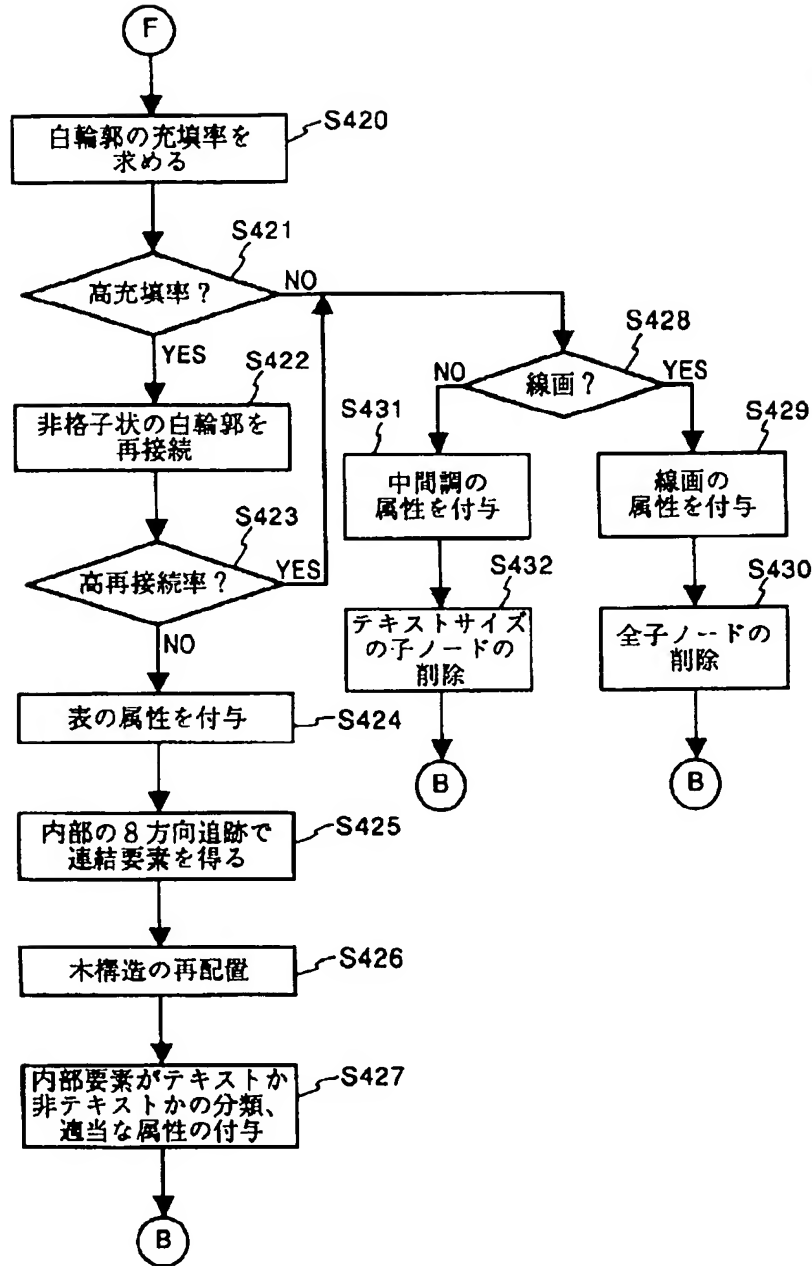
```

extern float *vector();
extern unsigned char *cvector();
extern int *ivector();
extern double *dvector();
extern unsigned char **cmatrix();
extern float **matrix();
extern int **imatrix();
extern double **dmatrix();
extern void free_vector();
extern void free_ivector();
extern void free_dvector();
extern void free_matrix();
extern void free_imatrix();
extern void free_dmatrix();
  
```

【図4B】



【図4C】



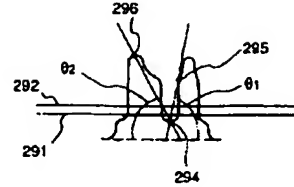
【図26A】

Satisfy

【図34B】

Satisfy

【図29B】



【図34A】



【図163】

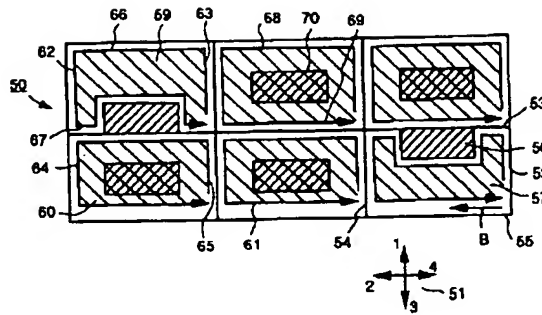
```

HISTOGRAM_THRESHOLD;
charon = charon->next; // Get the next character
}
free_vector(histogram); // Deallocate the memory

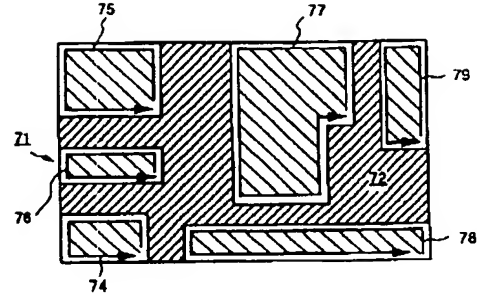
```

【図7A】

\* \*



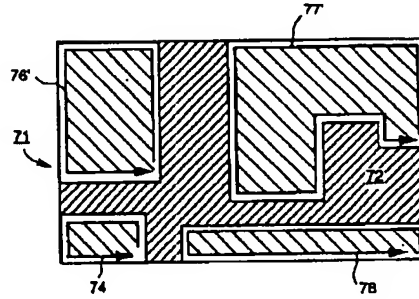
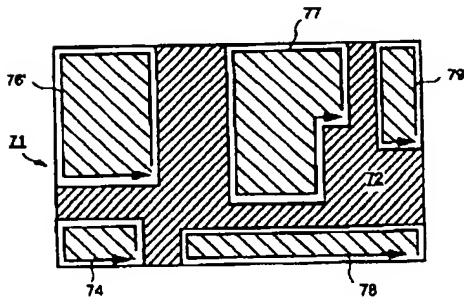
【図7B】



【図7C】

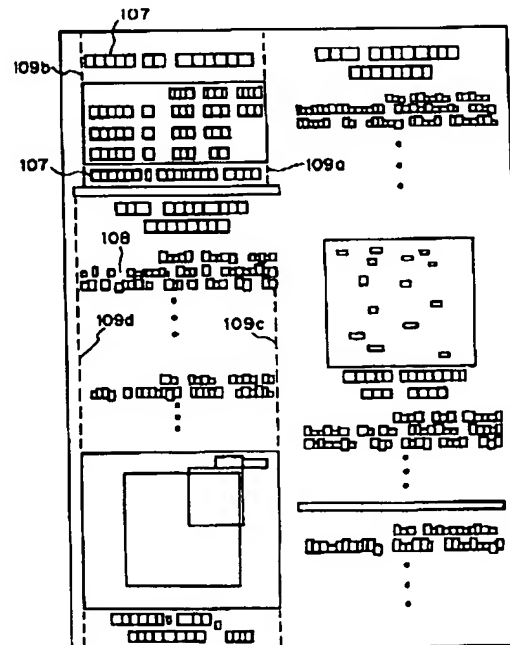
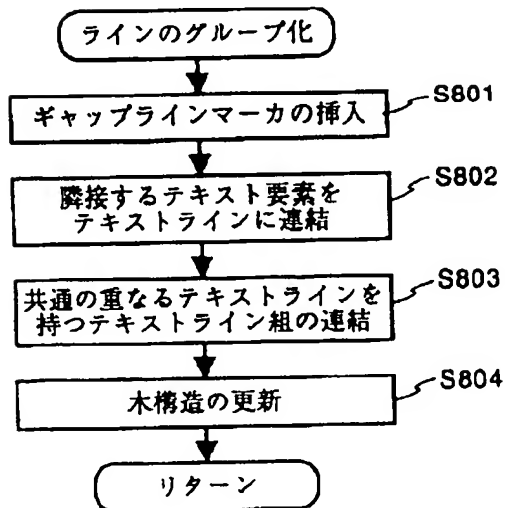
※ ※

【図7D】



【図8】

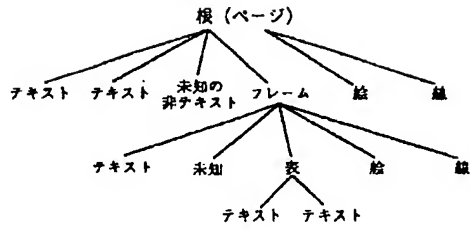
【図11】





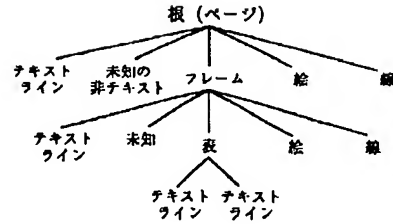


【図14】

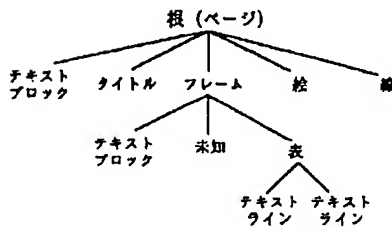


\* \*

【図15】

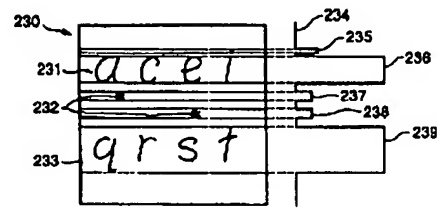


【図16】



※ ※

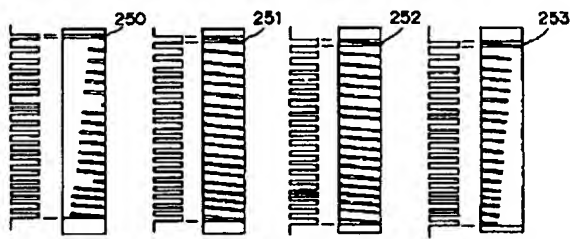
【図18A】



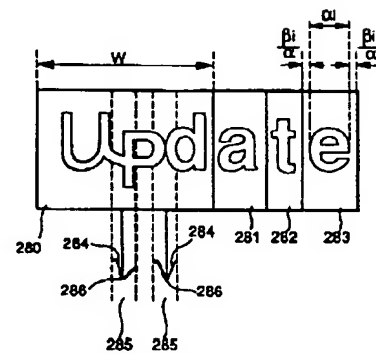
【図19C】

★ ★

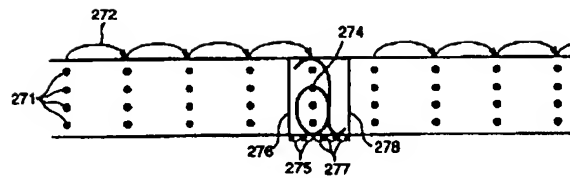
【図24】



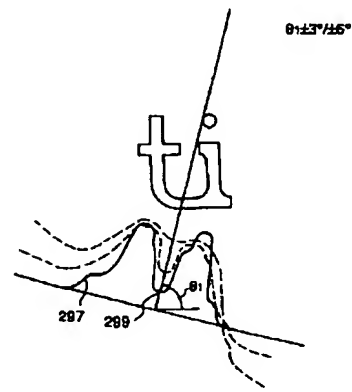
【図22】



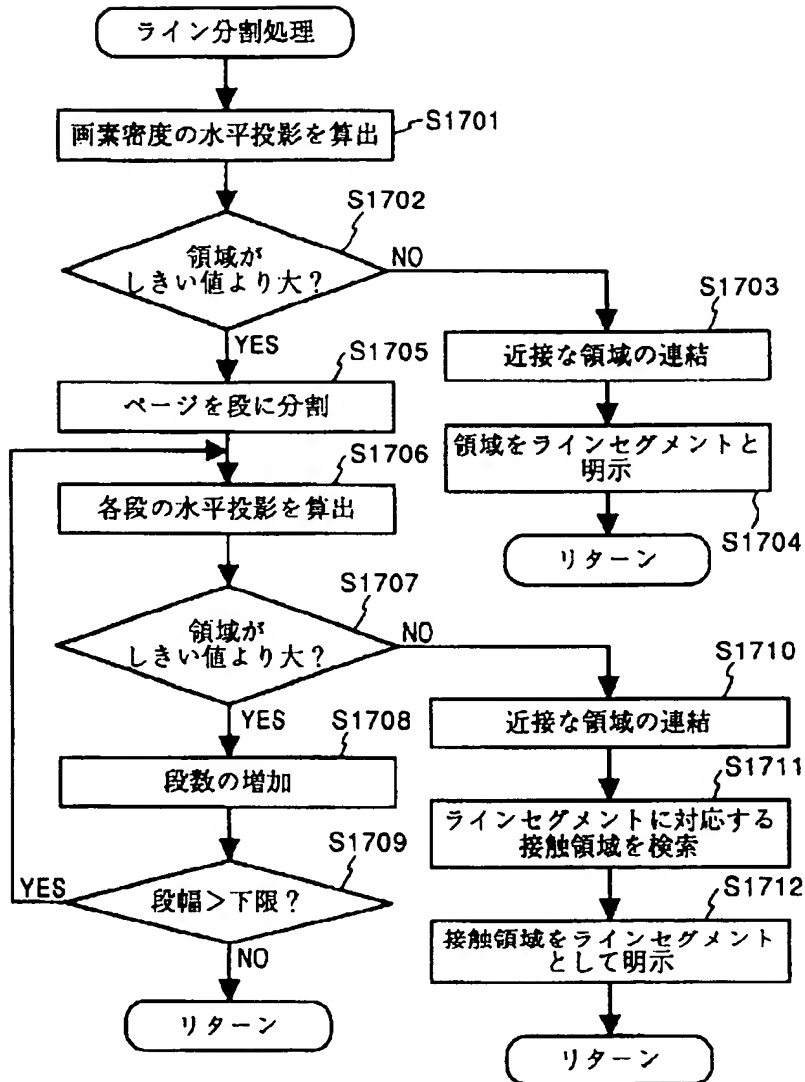
【図29C】



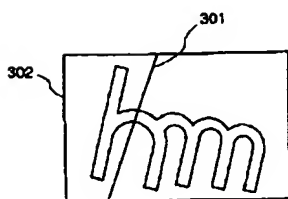
【図29D】



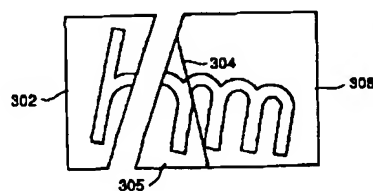
【図17】



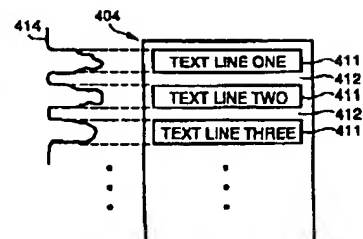
【図31A】



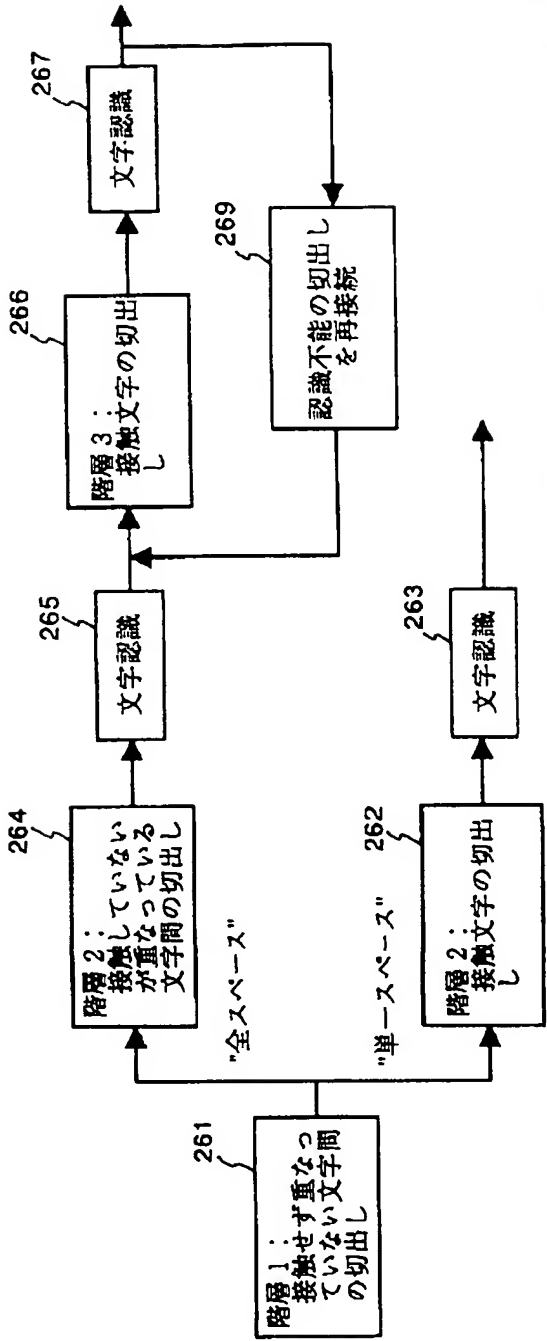
【図31B】



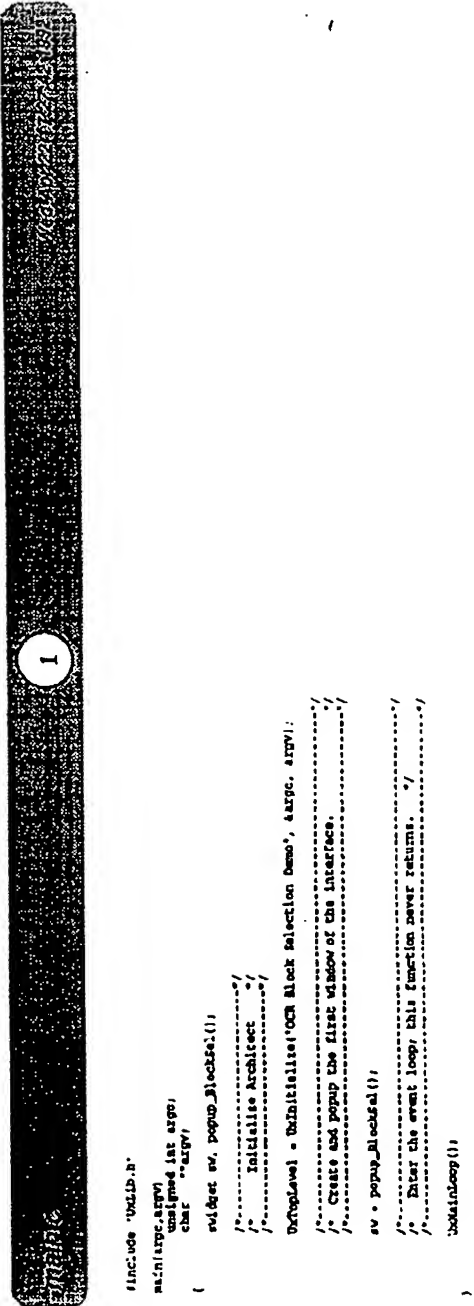
【図33A】



【図20】



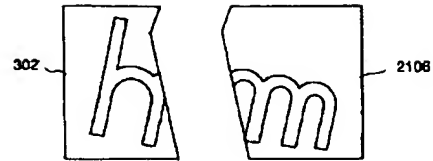
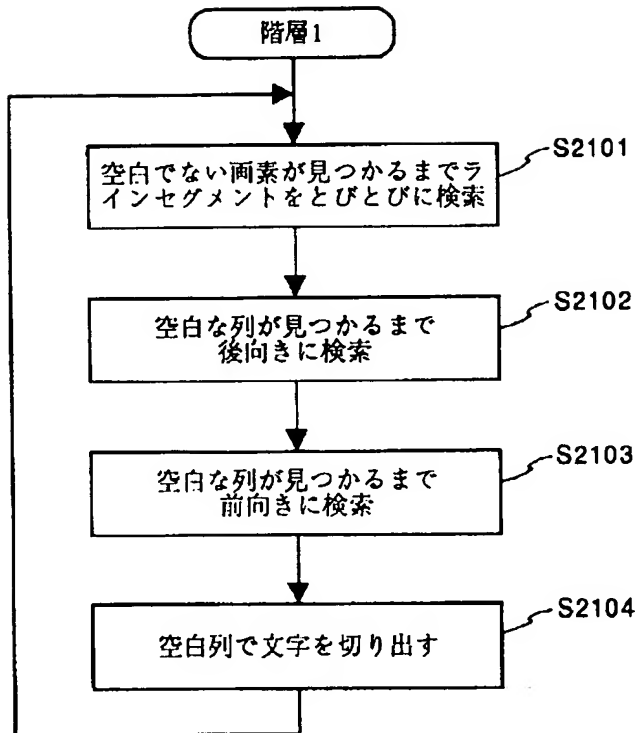
【図36】



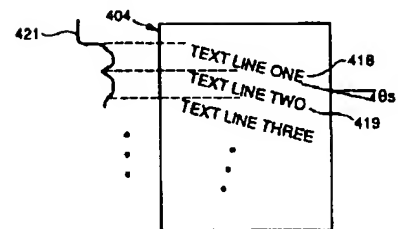
【図21】

\* \*

【図31C】

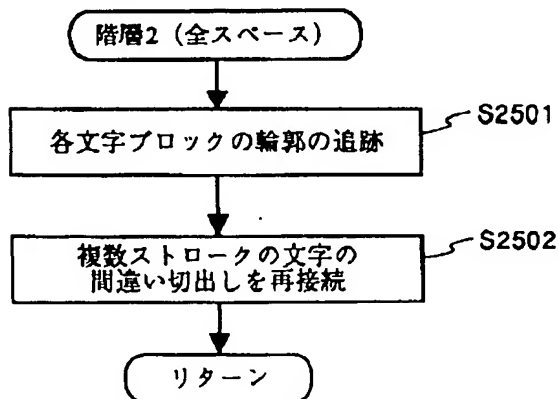


【図33B】



【図25】

【図32】

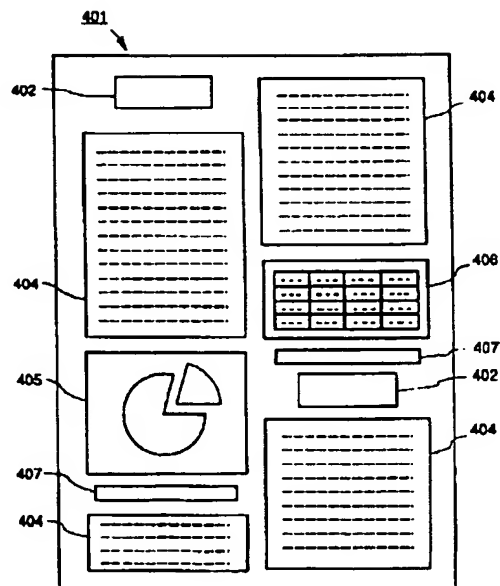


【図129】

```

struct sectionblock "previous;
struct sectionblock "next;

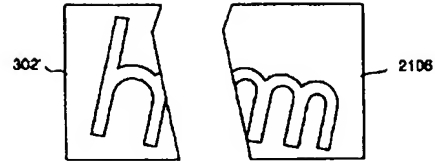
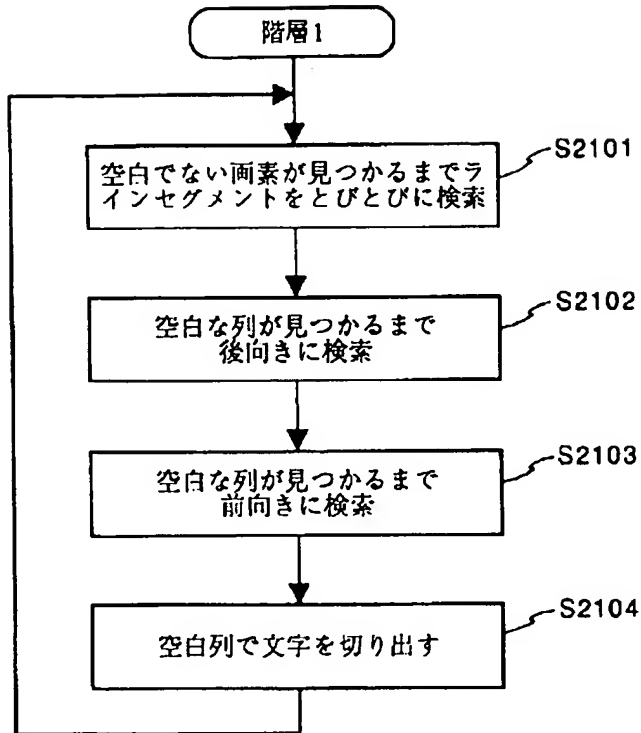
int upper_y, lower_y, left_x, right_x;
int width, length;
int index;
int multilineblock;
int att;
int boundary_index;
};
  
```



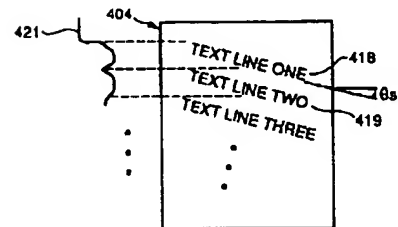
【図21】

\* \*

【図31C】

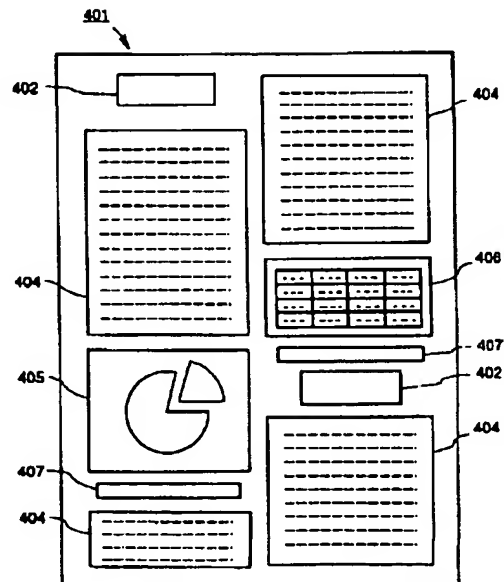
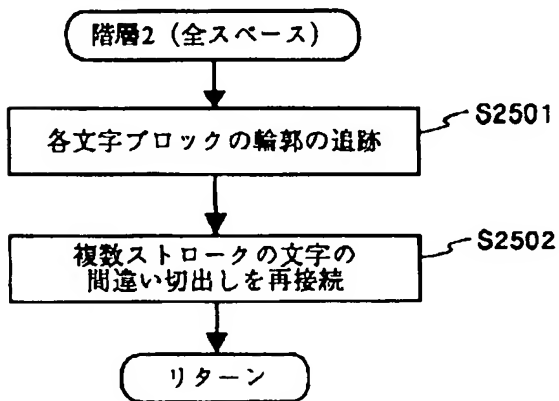


【図33B】



【図25】

【図32】

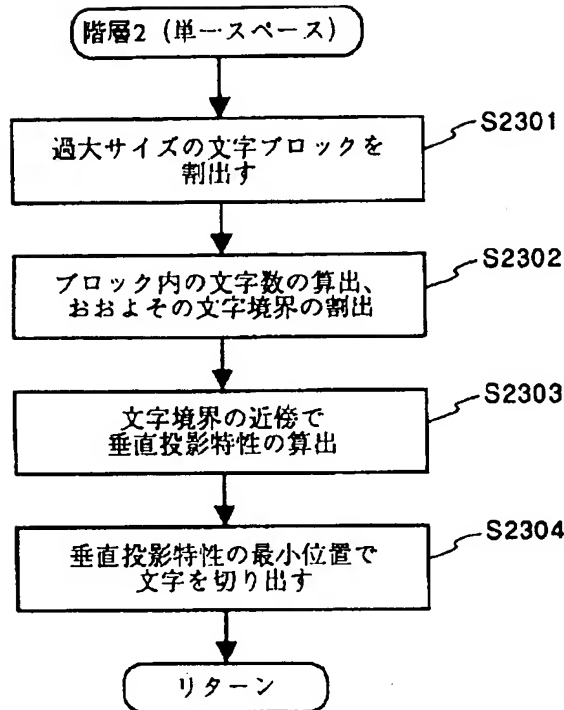


【図129】

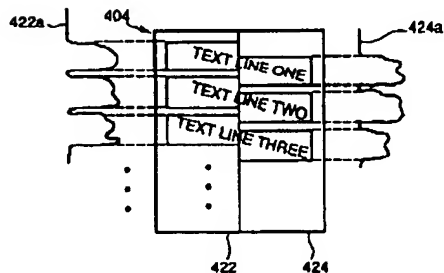
```

struct sectionblock "previous;
struct sectionblock "next;
int upper_y, lower_y, left_x, right_x;
int width, length;
int index;
int runlineblock;
int att;
int boundary_index;
};
  
```

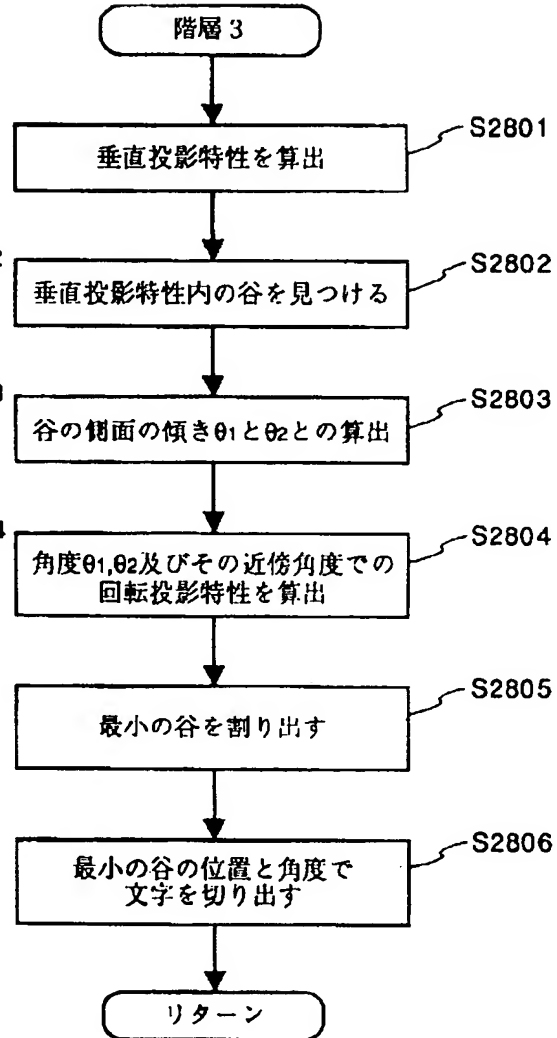
【図23】



【図33C】



【図28】



【図14】

```

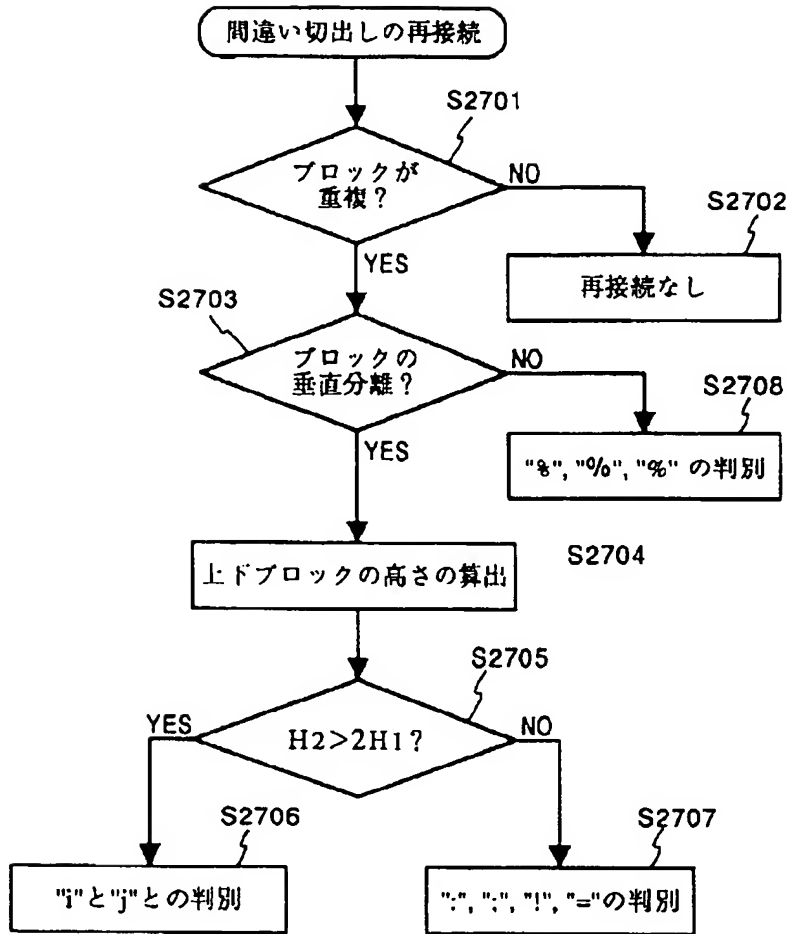
/* ARGSUSED */
static void okCallback_QuitMessageBox(UxWidget, UxContext, UxCallbackArg)
{
    UxWidget    UxWidget;
    _UxQuitDialog  *UxContext;
    int           UxCallbackArg;

    UxWidget    UxThisWidget;
    UxThisWidget= UxWidgetToUxWidget(UxWidget);
    _UxFromContext(UxContext);

    {
        exit(1);
    }

    _UxToContext(UxContext);
}
  
```

【図27】



【図46】

```

/*
 * _UnCon->DisplayForm= DisplayForm;
 * _UnCon->DisplayFormClose_Fnc= DisplayFormClose_Fnc;
 * _UnCon->DisplayForm_DrawArea= DisplayForm_DrawArea;
 */
static void _UnFromContext(_UnCon)
{
    _UnCon->DisplayForm = DisplayForm;
    _UnCon->DisplayFormClose_Fnc = DisplayFormClose_Fnc;
    _UnCon->DisplayForm_DrawArea = DisplayForm_DrawArea;
}

/* ARGUSED */
static void activateCallback_DisplayFormClose_Fnc(Widget, UnContext, UnCallbackArg)
{
    Widget widget;
    _UnConDisplayForm = UnContext;
    int UnCallbackArg;
    {
        widget = UnThisWidget;
        UnThisWidget = UnWidgetToWidget(UnWidget);
        _UnFromContext(UnContext);
        {
            UnPopdownInterface(DisplayForm);
        }
    }
    _UnToContext(UnContext);
}

```

【図133】

```

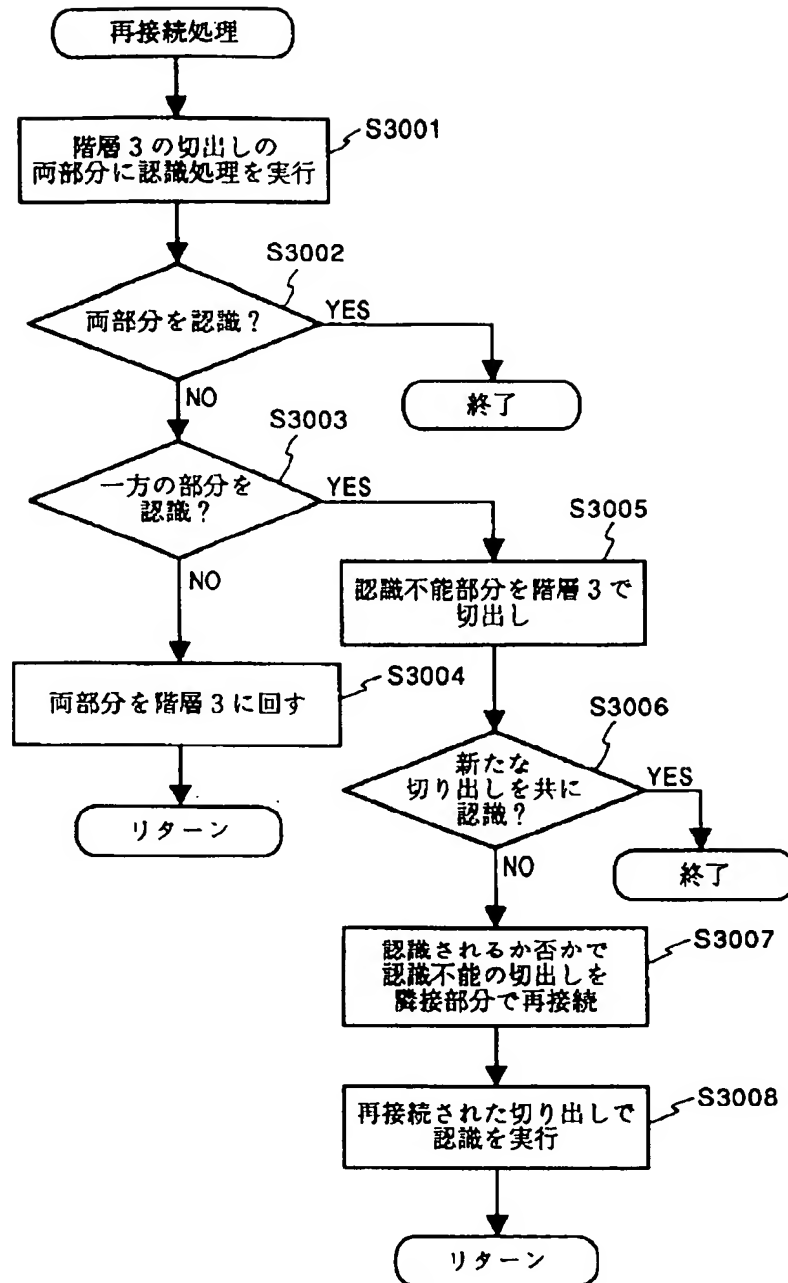
#define X 0
#define G 1
#define B 2

int color_comp[3][3]={{0x45, 0xcd, 0x00},
                      {0xf0, 0x80, 0x00},
                      {0xf0, 0x00, 0xf0},
                      {0xf0, 0xf0, 0x00},
                      {0xf0, 0xf0, 0x00},
                      {0x45, 0xcd, 0x00},
                      {0xf0, 0x80, 0x00},
                      {0xf0, 0x00, 0xf0},
                      {0x4b, 0x4e, 0x5a}};

/*
int color_comp[3][3]={{0x00, 0xc0, 0x00},
                      {0x00, 0xc0, 0x00},
                      {0x00, 0xc0, 0x00},
                      {0x00, 0xc0, 0x00},
                      {0x00, 0xc0, 0x00},
                      {0x00, 0xc0, 0x00},
                      {0x00, 0xc0, 0x00},
                      {0x00, 0xc0, 0x00},
                      {0x00, 0xc0, 0x00}};
*/

```

【図30】





【例 37】

[illegible]



—245—

[illegible]





【図42】

```

_UxCon->FileSelectionDialog= FileSelectionDialog;
_UxCon->FileSelectionBox= FileSelectionBox;
_UxCon->parent= parent;
}
static void _UxFromContext(_UxCon)
_UxCFileSelectionDialog *_UxCon;
{
    FileSelectionDialog= _UxCon->FileSelectionDialog;
    FileSelectionBox= _UxCon->FileSelectionBox;
    parent= _UxCon->parent;
}
/* ARGSUSED */
static void cancelCallback_FileSelectionBox(UxWidget, UxContext, UxCallbackArg)
Widget UxWidget;
_UxCFileSelectionDialog *_UxContext;
int UxCallbackArg;
{
    sWidget UxThisWidget;
    UxThisWidget= UxWidgetToWidget(UxWidget);
    _UxFromContext(UxContext);
    {
        UxPopdownInterface(FileSelectionDialog);
    }
    _UxToContext(UxContext);
}
/* ARGSUSED */
static void okCallback_FileSelectionBox(UxWidget, UxContext, UxCallbackArg)
Widget UxWidget;
_UxCFileSelectionDialog *_UxContext;
int UxCallbackArg;
{
    sWidget UxThisWidget;
    UxThisWidget= UxWidgetToWidget(UxWidget);
    _UxFromContext(UxContext);
    {
        handle_tiff_handle;
        UxPopdownInterface(FileSelectionDialog);
        strcpy(filename, get_input_filename());
        printf("Open %s\n", filename);
        tiff_handle = UxCreateSubproc ("usr/local/bin/X11/imageview", filename,
UxAppendTo);
        if (tiff_handle == -1) {
            error_create_subproc ();
            return;
        }
        if (UxRunSubproc (tiff_handle, NULL) == -1) {
            error_run_subproc ();
            return;
        }
        open_tiff_file(filename);
    }
    _UxToContext(UxContext);
}

```

【図48】

```

}
static void _UxFromContext(_UxCon)
_UxCInfoForm *_UxCon;
{
    InfoForm= _UxCon->InfoForm;
    InfoFormClose_FBG= _UxCon->InfoFormClose_FBG;
    InfoFormArea= _UxCon->InfoFormArea;
}
/* ARGSUSED */
static void activateCallback_InfoFormClose_FBG(UxWidget, UxContext, UxCallbackArg)
Widget UxWidget;
_UxCInfoForm *_UxContext;
int UxCallbackArg;
{
    sWidget UxThisWidget;
    UxThisWidget= UxWidgetToWidget(UxWidget);
    _UxFromContext(UxContext);
    {
        UxPopdownInterface (InfoForm);
    }
    _UxToContext(UxContext);
}

```

【図62】

```

int width, height;
{
    int x, y, xdis, ydis, i;
    static fid = 33;
    init_graphics(wd);
    xs_clearwin(wd);
    xdis = (height-30)/(TOTALCOLOR);
    xs_getfont(wd);
    for (i = 0; i < TOTALCOLOR; i++){
        xs_set_foreground(color_code[i], wd);
        xs_setfont(wd, fid);
        xs_string(wd, 30, (i+1)*xdis+10, color_label[i]);
    }
    xs_flush(wd);
}

```

—249—

[illegible]

[illegible]





—252—

8

-253-

【図51】

```

        }
        *height = win_height;
    }

    xs_draw_line(wd, x1, y1, x2, y2);
    widget wd;
    int x1, y1, x2, y2;
    }
    xs_create_line(widget(wd), xwindow(wd), drawing_gc, x1, y1, x2, y2);
}

xs_color_paint(wd, x, y, color)
    widget wd;
    int x, y, color;
{
    Arg args[1];
    int foreground;
    XSetArg(args[0], XInt, foreground, &foreground);
    XSetValues(wd, args, 1);
    xs_set_foreground(widget(wd), drawing_gc, color);
    xs_color_paint(widget(wd), xwindow(wd), drawing_gc, x, y);
    xs_set_foreground(widget(wd), drawing_gc, foreground);
}

xs_color_line(wd, x1, y1, x2, y2, color)
    widget wd;
    int x1, y1, x2, y2, color;
{
    Arg args[1];
    int foreground;
    XSetArg(args[0], XInt, foreground, &foreground);
    XSetValues(wd, args, 1);
    xs_set_foreground(widget(wd), drawing_gc, color);
    xs_create_line(widget(wd), xwindow(wd), drawing_gc, x1, y1, x2, y2);
    xs_set_foreground(widget(wd), drawing_gc, foreground);
}

xs_color_box(wd, x1, y1, x2, y2, color)
    widget wd;
    int x1, y1, x2, y2, color;
{
    Arg args[1];
    int foreground;
    XSetArg(args[0], XInt, foreground, &foreground);
    XSetValues(wd, args, 1);
    xs_set_foreground(widget(wd), drawing_gc, color);
    xs_create_box(widget(wd), xwindow(wd), drawing_gc, x1, y1, x2, y2);
    xs_set_foreground(widget(wd), drawing_gc, foreground);
}

xs_color_area(wd, x, y, str, color)
    widget wd;
    int x, y, color;
    char *str;
{
    Arg args[1];
    int foreground;
    XSetArg(args[0], XInt, foreground, &foreground);
    XSetValues(wd, args, 1);
}

```

```

        xs_set_foreground(widget(wd), drawing_gc, color);
        xs_create_line(widget(wd), xwindow(wd), drawing_gc, x, y, str, strlen(str));
        xs_set_foreground(widget(wd), drawing_gc, foreground);
    }

    static char **font;
    XFontSet font(wd);
    widget wd;
    {
        int i, num = $3;
        font = XListFonts(widget(wd), "...", 30, &num);
        printf("num = %d\n", num);
    }
    xs_set_font(wd, fid);
    widget wd;
    int fid;
    {
        XFontStruct *font_info;
        if ((font_info = XLoadQueryFont(widget(wd), font(fid)) != NULL))
        {
            printf("num not open the font fid = %d\n", fid);
            exit(-1);
        }
        xs_set_font(widget(wd), drawing_gc, font_info->fid);
        printf("fid = %d\n", fid);
    }
}

```

```

/*
int getbyte(c, file_position, eoferror)
FILE *f;
long int *file_position;
int *eoferror;
{
    int c;
    if (!getc(f),
        if ((c == EOF) && (*eoferror == 1))
            printf("\n Error, Premature End of File :in getbyte()\n");
        )
        exit(1);
        return c;
}

/* Read in an integer from a file. (LSD, MSB)
FILE *f;
long int *getbytes(int sp)
{
    long int *ip;
    int c;
    long inc c;
    c = getbyte(f, sp, 1);
    c = getbyte(f, sp, 1);
    c = (long int)(c + c*256);
    return c;
}

/* Read in a long integer
FILE *f;
long int *getbytes(long sp)
{
    long int *ip;
    long int c;
    c = getbyte(f, sp, 1);
    c = getbyte(f, sp, 1);
    return c + c*256*256;
}

/* Read in part of a Tiff file
.....
struct images *readtiff(filename)
char *filename;
{
    FILE *tagfile;
    long int tagfile_position;
    long int width;
    long int height;
    long int black;
    long int bps;
    long int raster_offset;
    long int nuntage, tag_type, len, value; /* Tag values
    long int i;
    int v;
    struct images *img;
    if (tagfile = fopen(filename, "rb")) == NULL
*/
#include "blockgeneral.h"
#include images.h
#include chanfile.h

extern unsigned char **cmatrix[];

void images_allocate(page)
struct images *page;
{
    if ((page->size != 0) && (page->sizec != 0))
        free_matrix((page->pxlen), 0, page->size-1, 0, page->sizec-1);
    page->sizea = 0;
    page->sizec = 0;
}

/* Allocate memory for an image from memory
.....
struct images *image_allocate(c)
int c;
{
    struct images *page;
    page = (struct images *)malloc(sizeof(struct images));
    if (page == NULL) {
        printf("Allocation error, in image_allocate\n");
        exit(1);
    }
    if (c <= 0 || c >= 0)
    {
        printf("Illegal image size, td by %d \n", c);
        exit(1);
    }
    /* Set image sizes
    page->sizea = c;
    page->sizec = c;
    page->pixa = matrix[0, page->sizea-1, 0, page->sizec-1];
    return page;
}

/* Allocate memory for an image from memory
.....
void images_deallocate(page)
struct images *page;
{
    if ((page->sizea != 0) && (page->sizec != 0))
        free_matrix((page->pxlen), 0, page->sizea-1, 0, page->sizec-1);
    page->sizea = 0;
    page->sizec = 0;
}

/* Read in a byte from a tiff file
.....
if (tagfile = fopen(filename, "rb")) == NULL

```

—256—

```

print(stderr, "Error in opening %s\n", filename);
exit(1);
}

tagfile_position = 0;

/* Read in header and make sure it's correct */
/* getHeaderTagfilePosition() */
/* print("Header tag = %d\n", tag); */
if (tag != 0x4d49)
{
    printf("... Error: TIFF file not in Intel Format\n");
    exit(1);
}

/* Read in Magic number */
/* getBytes(tagfile, tagfile_position); */
/* print("i = %x\n", i); */
if (i != 0x002a)
{
    printf("BAD magic number for TIFF file\n");
    exit(1);
}

/* Read First File Directory */
/* getBytes(tagfile, tagfile_position); */
/* print("tagfile_position = %d\n", tagfile_position); */
i = getBytes(tagfile, tagfile_position);
}

/* Then in Directory */
/* print("numtags = %d\n", numtags); */
while (numtags == 0)
{
    tag = getBytes(tagfile, tagfile_position);
    len = getBytes(tagfile, tagfile_position);
    value = getBytes(tagfile, tagfile_position);
    if ((len != 1) && ((tag & 0x01) == 0))
    {
        printf("Error: Tag Length is 1\n");
        exit(1);
    }

    /* print("tag = %x\n", tag); */
    switch (tag)
    {
        case 0x0000: /* NewSubFileType */
        {
            printf("Can't Deal with Non-Standard NewSubFileType\n");
            exit(1);
        }
        break;
        case 0x0001: /* NewSubFileType */
        {
            printf("Subfile Type not full resolution\n");
            exit(1);
        }
        break;
        case 0x0002: /* Width */
        {
            width = value;
        }
        break;
        case 0x0003: /* Length */
        {
            length = value;
        }
        break;
    }
}

print(stderr, "Error: Bits/sample must be 1\n");
exit(1);
}

/* Bits per sample */
case 0x0010:
{
    if (value != 1)
    {
        printf("Error: Bits/sample must be 1\n");
        exit(1);
    }
    break;
}
case 0x0011: /* Compression */
{
    if (value != 1)
    {
        printf("Compression Used! Error\n");
        exit(1);
    }
    break;
}
case 0x0012: /* Black is */
{
    break;
}
case 0x0013: /* Thresholding tag */
{
    if ((value & 1) != (value > 3))
    {
        printf("Thresholding Tag = Illegal value\n");
        exit(1);
    }
    break;
}
case 0x0014: /* Offset to raster data */
{
    raster_offset = value;
    break;
}
case 0x0015: /* Bits = value */
{
    break;
}
case 0x0016: /* Rows */
{
    break;
}
case 0x0017: /* Columns */
{
    break;
}
case 0x0018: /* Rows */
{
    break;
}
case 0x0019: /* Columns */
{
    break;
}
case 0x001a: /* Rows */
{
    break;
}
case 0x001b: /* Columns */
{
    break;
}
case 0x001c: /* Rows */
{
    break;
}
case 0x001d: /* Columns */
{
    break;
}
case 0x001e: /* Rows */
{
    break;
}
case 0x001f: /* Columns */
{
    break;
}
case 0x0020: /* Rows */
{
    break;
}
case 0x0021: /* Columns */
{
    break;
}
case 0x0022: /* Rows */
{
    break;
}
case 0x0023: /* Columns */
{
    break;
}
case 0x0024: /* Rows */
{
    break;
}
case 0x0025: /* Columns */
{
    break;
}
case 0x0026: /* Rows */
{
    break;
}
case 0x0027: /* Columns */
{
    break;
}
case 0x0028: /* Rows */
{
    break;
}
case 0x0029: /* Columns */
{
    break;
}
case 0x002a: /* Rows */
{
    break;
}
case 0x002b: /* Columns */
{
    break;
}
case 0x002c: /* Rows */
{
    break;
}
case 0x002d: /* Columns */
{
    break;
}
case 0x002e: /* Rows */
{
    break;
}
case 0x002f: /* Columns */
{
    break;
}
case 0x0030: /* Rows */
{
    break;
}
case 0x0031: /* Columns */
{
    break;
}
case 0x0032: /* Rows */
{
    break;
}
case 0x0033: /* Columns */
{
    break;
}
case 0x0034: /* Rows */
{
    break;
}
case 0x0035: /* Columns */
{
    break;
}
case 0x0036: /* Rows */
{
    break;
}
case 0x0037: /* Columns */
{
    break;
}
case 0x0038: /* Rows */
{
    break;
}
case 0x0039: /* Columns */
{
    break;
}
case 0x003a: /* Rows */
{
    break;
}
case 0x003b: /* Columns */
{
    break;
}
case 0x003c: /* Rows */
{
    break;
}
case 0x003d: /* Columns */
{
    break;
}
case 0x003e: /* Rows */
{
    break;
}
case 0x003f: /* Columns */
{
    break;
}
case 0x0040: /* Rows */
{
    break;
}
case 0x0041: /* Columns */
{
    break;
}
case 0x0042: /* Rows */
{
    break;
}
case 0x0043: /* Columns */
{
    break;
}
case 0x0044: /* Rows */
{
    break;
}
case 0x0045: /* Columns */
{
    break;
}
case 0x0046: /* Rows */
{
    break;
}
case 0x0047: /* Columns */
{
    break;
}
case 0x0048: /* Rows */
{
    break;
}
case 0x0049: /* Columns */
{
    break;
}
case 0x004a: /* Rows */
{
    break;
}
case 0x004b: /* Columns */
{
    break;
}
case 0x004c: /* Rows */
{
    break;
}
case 0x004d: /* Columns */
{
    break;
}
case 0x004e: /* Rows */
{
    break;
}
case 0x004f: /* Columns */
{
    break;
}
case 0x0050: /* Rows */
{
    break;
}
case 0x0051: /* Columns */
{
    break;
}
case 0x0052: /* Rows */
{
    break;
}
case 0x0053: /* Columns */
{
    break;
}
case 0x0054: /* Rows */
{
    break;
}
case 0x0055: /* Columns */
{
    break;
}
case 0x0056: /* Rows */
{
    break;
}
case 0x0057: /* Columns */
{
    break;
}
case 0x0058: /* Rows */
{
    break;
}
case 0x0059: /* Columns */
{
    break;
}
case 0x005a: /* Rows */
{
    break;
}
case 0x005b: /* Columns */
{
    break;
}
case 0x005c: /* Rows */
{
    break;
}
case 0x005d: /* Columns */
{
    break;
}
case 0x005e: /* Rows */
{
    break;
}
case 0x005f: /* Columns */
{
    break;
}
case 0x0060: /* Rows */
{
    break;
}
case 0x0061: /* Columns */
{
    break;
}
case 0x0062: /* Rows */
{
    break;
}
case 0x0063: /* Columns */
{
    break;
}
case 0x0064: /* Rows */
{
    break;
}
case 0x0065: /* Columns */
{
    break;
}
case 0x0066: /* Rows */
{
    break;
}
case 0x0067: /* Columns */
{
    break;
}
case 0x0068: /* Rows */
{
    break;
}
case 0x0069: /* Columns */
{
    break;
}
case 0x006a: /* Rows */
{
    break;
}
case 0x006b: /* Columns */
{
    break;
}
case 0x006c: /* Rows */
{
    break;
}
case 0x006d: /* Columns */
{
    break;
}
case 0x006e: /* Rows */
{
    break;
}
case 0x006f: /* Columns */
{
    break;
}
case 0x0070: /* Rows */
{
    break;
}
case 0x0071: /* Columns */
{
    break;
}
case 0x0072: /* Rows */
{
    break;
}
case 0x0073: /* Columns */
{
    break;
}
case 0x0074: /* Rows */
{
    break;
}
case 0x0075: /* Columns */
{
    break;
}
case 0x0076: /* Rows */
{
    break;
}
case 0x0077: /* Columns */
{
    break;
}
case 0x0078: /* Rows */
{
    break;
}
case 0x0079: /* Columns */
{
    break;
}
case 0x007a: /* Rows */
{
    break;
}
case 0x007b: /* Columns */
{
    break;
}
case 0x007c: /* Rows */
{
    break;
}
case 0x007d: /* Columns */
{
    break;
}
case 0x007e: /* Rows */
{
    break;
}
case 0x007f: /* Columns */
{
    break;
}
case 0x0080: /* Rows */
{
    break;
}
case 0x0081: /* Columns */
{
    break;
}
case 0x0082: /* Rows */
{
    break;
}
case 0x0083: /* Columns */
{
    break;
}
case 0x0084: /* Rows */
{
    break;
}
case 0x0085: /* Columns */
{
    break;
}
case 0x0086: /* Rows */
{
    break;
}
case 0x0087: /* Columns */
{
    break;
}
case 0x0088: /* Rows */
{
    break;
}
case 0x0089: /* Columns */
{
    break;
}
case 0x008a: /* Rows */
{
    break;
}
case 0x008b: /* Columns */
{
    break;
}
case 0x008c: /* Rows */
{
    break;
}
case 0x008d: /* Columns */
{
    break;
}
case 0x008e: /* Rows */
{
    break;
}
case 0x008f: /* Columns */
{
    break;
}
case 0x0090: /* Rows */
{
    break;
}
case 0x0091: /* Columns */
{
    break;
}
case 0x0092: /* Rows */
{
    break;
}
case 0x0093: /* Columns */
{
    break;
}
case 0x0094: /* Rows */
{
    break;
}
case 0x0095: /* Columns */
{
    break;
}
case 0x0096: /* Rows */
{
    break;
}
case 0x0097: /* Columns */
{
    break;
}
case 0x0098: /* Rows */
{
    break;
}
case 0x0099: /* Columns */
{
    break;
}
case 0x009a: /* Rows */
{
    break;
}
case 0x009b: /* Columns */
{
    break;
}
case 0x009c: /* Rows */
{
    break;
}
case 0x009d: /* Columns */
{
    break;
}
case 0x009e: /* Rows */
{
    break;
}
case 0x009f: /* Columns */
{
    break;
}
case 0x00a0: /* Rows */
{
    break;
}
case 0x00a1: /* Columns */
{
    break;
}
case 0x00a2: /* Rows */
{
    break;
}
case 0x00a3: /* Columns */
{
    break;
}
case 0x00a4: /* Rows */
{
    break;
}
case 0x00a5: /* Columns */
{
    break;
}
case 0x00a6: /* Rows */
{
    break;
}
case 0x00a7: /* Columns */
{
    break;
}
case 0x00a8: /* Rows */
{
    break;
}
case 0x00a9: /* Columns */
{
    break;
}
case 0x00aa: /* Rows */
{
    break;
}
case 0x00ab: /* Columns */
{
    break;
}
case 0x00ac: /* Rows */
{
    break;
}
case 0x00ad: /* Columns */
{
    break;
}
case 0x00ae: /* Rows */
{
    break;
}
case 0x00af: /* Columns */
{
    break;
}
case 0x00b0: /* Rows */
{
    break;
}
case 0x00b1: /* Columns */
{
    break;
}
case 0x00b2: /* Rows */
{
    break;
}
case 
```

【図54】

```

if (v == EOF)
{
    printf("\n Error: Premature End of File found in tiff file\n");
    exit(1);
}
if ((v & 128) == black*128)
    page->pixel[i][j] = 0;
else
    page->pixel[i][j] = 1;
v = v << 1;
}
fclose(tagfile);
return PAGE;
}

```

【図86】

```

        release_block(tmp1);
        tmp1 = tmp11;
    }
    for (; tmp1 != NULL; ) {
        tmp12 = tmp1->next;
        release_block(tmp1);
        tmp1 = tmp12;
    }
    free_rectblock(initial_block);
    return;
}
/* Draw text components for display */
/* Draw text components for display */

void block_circumulate(p, width, height, p_ori)
unsigned char **p;
int width, height;
unsigned char **p_ori;
{
    int i, j;
    /* open file for storing block data */
    original_p = p_ori;
    reduce_factor = SCALE_RATIO;

    for (i = 0; i <= H; i++)
        for (j = 0; j <= W; j++)
            direct_status[i][j] = 0;

    block_file = fopen("block_file.dat", "w");

    text_length = 130.0/SCALE_RATIO;
    total_text = 0;

    /* the root block stands for the whole page image unit */
    root_rectblock = new_block(0,0);
    root_rectblock->width = width;
    root_rectblock->length = height;
    root_rectblock->upper_y = 0;
    root_rectblock->lower_y = height-1;
    root_rectblock->left_x = 0;
    root_rectblock->right_x = width-1;

    /* find blocks inside the page range */
    block_in_range(p, 0, 0, width-1, height-1, root_rectblock);
    printf("get out of block_in_range\n");
    rearrange_rectblock(root_rectblock);
    printf("get out of rearrange_rectblock\n");

    text_discriminate(root_rectblock);

    printf("get out of text_dis\n");

    firstlevelnontextsearch(p, root_rectblock);

    printf("get out of firstlevelnontextsearch\n");
    print_block(1, root_rectblock);

    fclose(block_file);
    printf("before get out of block_construct\n");
}

```

【図137】

## Source Code for Proportional Spaced Segmentation

【例 5 5】

[illegible]



—259—

[illegible]



【図58】

```

for (tap1 = firstblock->firstwhite, tap2 != NULL; ){
    /* printf("subblock\n");
    tap2 = tap1->nextwhite;
    if (tap2 == NULL) continue;
    for (i = 0; i < tap1->numsubblock; i++){
        free_ptr(tap1->subblock[i].ptrline);
        tap1->subblock[i].ptrline = tap2->subblock[i].ptrline;
        tap1->subblock[i].lower_j = tap2->subblock[i].lower_j;
    }
    free(struct whiteblock *tap1->subwhite);
    free_ptr(tap1);
    /* printf("after free white next = %d\n", tap2); */
    tap2 = tap1;
    /* printf("after release white\n"); */
    free_rectblock(firstblock);
}

/* .....
/* .....
/* .....
void free_section(firstsection)
struct sectionblock *firstsection
{
    struct sectionblock *tap1, *tap2;
    struct lineblock *tap1, *tap2;
    struct block_rect *tap1, *tap2;
    /* printf("get into free section\n");
    /* for (tap1 = firstsection->firstsubsection, tap1 != NULL; ){
        tap2 = tap1->next;
        printf("section = %d section = %d\n", tap1, tap1->index);
        for (tap1 = tap1->firstlineblock, tap1 != NULL; ){
            tap2 = tap1->next;
            /* printf("tap1 = %d line = %d first = %d current %d\n", tap1,
            tap1->index, tap1->first_block->index,
            tap1->current_block->index);
            /*
            for (tap1 = tap1->first_block, tap1 != NULL; ){
                tap2 = tap1->next;
                free_block(tap1);
            }
            printf("after free rect, tap2 = %d tap1 = %d\n", tap2, tap1);
            tap1 = NULL;
            tap2 = tap2;
            tap1->first_block = NULL;
            printf("free line tap1 = %d tap1->index = %d underold %d cap %d\n",
            tap1, tap1->index, tap1->under_section, tap1->of_section);
            /* free(struct lineblock *tap1);
            /* printf("after free line tap2 = %d\n", tap2);
            /*
}

```

【图 5-9】

```

0, reduce_image, width-1);

printf("before\n");
if (step==SP_COLOR){
    free_matrix(image, 0, reduce_image.height,
0, reduce_image.width-1);
    free_section(firstsectionblock);
    printf("after free section\n");
}

image_malloc(page);
step = 0;

printf("unreading....");
page = readfile(filename);
printf("Done, un");
get_ptr_of_upper(page==pixel, 1,1, page->size-1, page->size-1);
printf("link = %d size = %d un", page->size, page->size);
create_reduce_image();
step = SP_REDUCED;
}

/*..... get text block into text display
.....*/

void get_text(section, p, wd, color)
struct sectionblock *section;
unsigned char **p;
int color;
{
    struct lineblock *temp;
    struct block_root *temp2;
    int x, y, xl, yl;

    memset(section->color, wd);
    for (temp = section->first_block; temp != NULL; temp = temp->next){
        for (y = temp->upper_y; y < temp->lower_y; y++){
            for (x = temp->left_x; x < temp->right_x; x++){
                p[y][x] = TEXT_BLOCK;
            }
        }
        xl = (int)((float)x/section->displayform);
        yl = (int)((float)y/section->displayform);
        xl = (int)((float)x/section->scale_displayform);
        yl = (int)((float)y/section->scale_displayform);
        temp->block_root->temp2 = temp;
        temp->block_root->temp2 = temp;
    }
}

void get_subtext(section, p, wd)
struct sectionblock *section;
unsigned char **p;
int wd;
{
    int x, y, xl, yl;
}

/*..... get text into table text display
.....*/

void get_subtext(section, p, wd)
struct sectionblock *section;
unsigned char **p;
int wd;
{
    int x, y, xl, yl;
}

```

【図60】

```

        xs_set_foreground(TEXT_COLOR, wd);
        for (y = tmapsect->map_y; y < tmapsect->lower_y; y++)
            for (x = tmapsect->left_x; x < tmapsect->right_x; x++){
                if ((reduce_image_pixel(y)(x)(tmapsect->map_y)(x) < 0))
                    p[y][x] = TEXT_BLOCK;
                y1 = (int)((float)y/scale_displayform);
                x1 = (int)((float)x/scale_displayform);
                xs_picture(wd, x1, y1);
            }
    }

    /*..... get picture into picture display .....*/
    /*..... struct sectionblock "tmapsect2" .....*/
    void get_picture(tmapsect2, p, att, color, wd)
    struct sectionblock "tmapsect2";
    unsigned char *p;
    widget wd;
    {
        struct block_struct "tmapsect2";
        int x1, y1, x2, y2;
        xs_set_foreground(TEXT_COLOR, wd);
        for (y = tmapsect2->map_y; y < tmapsect2->lower_y; y++)
            for (x = tmapsect2->left_x; x < tmapsect2->right_x; x++){
                if ((reduce_image_pixel(y)(x)(tmapsect2->map_y)(x) < 0))
                    p[y][x] = att;
                y1 = (int)((float)y/scale_displayform);
                x1 = (int)((float)x/scale_displayform);
                xs_picture(wd, x1, y1);
            }
    }

    int x, y, x1, y1;
    xs_set_foreground(TEXT_COLOR, wd);
    for (y = tmapsect->map_y; y < tmapsect->lower_y; y++)
        for (x = tmapsect->left_x; x < tmapsect->right_x; x++){
            if ((reduce_image_pixel(y)(x)(tmapsect->map_y)(x) < 0))
                p[y][x] = att;
            y1 = (int)((float)y/scale_displayform);
            x1 = (int)((float)x/scale_displayform);
            xs_picture(wd, x1, y1);
        }
    }

    /*..... separate large .....*/
    /*..... void separate_image(firstsection, wd) .....*/
    struct sectionblock "firstsection";
    widget wd;
    {
        struct sectionblock "tmapsect", "tmapsect2", "tmapsect3";
        int x, y;

        for (tmapsect = firstsection->firstsection, tmapsect1 = NULL;
            get_text(tmapsect, h_image, wd, TEXT_COLOR);
            for (tmapsect = firstsection->firstsection; tmapsect != NULL;
                tmapsect = tmapsect->next){
                if (tmapsect->att != TEXT){
                    get_text(tmapsect, h_image, wd, TEXT_COLOR);
                    if (tmapsect->att != TEXT){
                        draw_line(tmapsect, wd, LINE_COLOR, 1);
                    } else if (tmapsect->att != NONE){
                        draw_line(tmapsect, wd, LINE_COLOR, 3);
                    } else if (tmapsect->att != VOID){
                        draw_line(tmapsect, wd, LINE_COLOR, 2);
                    } else if (tmapsect->att != NONE){
                        draw_line(tmapsect, wd, LINE_COLOR, 4);
                    }
                }
            }
    }

    /* print("get out line picture\n"); */

    /*..... get picture into picture display .....*/
    void get_line(tmapsect, p, att, color, wd)
    struct sectionblock "tmapsect";
    unsigned char *p;
    widget wd;
    {
        int x, y, x1, y1;
        xs_set_foreground(TEXT_COLOR, wd);
        for (y = tmapsect->map_y; y < tmapsect->lower_y; y++)
            for (x = tmapsect->left_x; x < tmapsect->right_x; x++){
                if ((reduce_image_pixel(y)(x)(tmapsect->map_y)(x) < 0))
                    p[y][x] = att;
            }
    }

```

【图 6-1】

[illegible]



【図64】

```

)
/* allocate a double matrix with rows (nr1..nrh) [nc1..nc2]. ~ /
double **matrix(nr1, nrh, nc1, nc2)
{
    int nr1, nrh, nc1, nc2;

    int i;
    double **m;
    m = (double **) malloc((nrh-nr1+1)*sizeof(double *));
    if (!m) perror("allocation failure 1 in matrix()");
    m += nr1;
    for (i = nr1; i <= nrh; i++) {
        if (!m[i]) malloc((nc2-nc1+1)*sizeof(double));
        if (!m[i]) perror("allocation failure 2 in matrix()");
        m[i] += nc1;
    }
    return m;
}

/* deallocate a float vector */
void free_vectorfv(float *, nh)
{
    int ni, nh;
    free((float *) (v+nh));
}

/* deallocate a int vector */
void free_vectoriv(int *, nh)
{
    int *, ni, nh;
    free((int *) (v+nh));
}

/* deallocate a double vector */
void free_vectordv(double *, nh)
{
    int ni, nh;
    free((double *) (v+nh));
}

/* deallocate a float matrix */
void free_matrixfv(float *, nrh, nc1, nc2)
{
    int nr1, nrh, nc1, nc2;

    int i;
    for (i = nrh; i >= nr1; i--) free((float *) (m[i]-nc1));
    free((float *) (m+nr1));
}

/* deallocate an int matrix */
void free_matrixiv(int *, nrh, nc1, nc2)
{
    int nr1, nrh, nc1, nc2;

    int i;
    for (i = nrh; i >= nr1; i--) free((int *) (m[i]-nc1));
    free((int *) (m+nr1));
}

```



—267—

—267—

【図66】

```

    struct data_outline *new_data_outline(n1, n2)
    int n1, n2;
    {
        int i;
        struct data_outline *v;
        v = (struct data_outline *) malloc(sizeof(struct data_outline));
        if (!v) {
            printf("allocation error: data_outline\n");
            exit(1);
        }
        v = v+1;
        for (i = n1; i <= n2; i++) {
            v[i].total_left = v[i].total_right = 0;
            v[i].current_left = v[i].current_right = NULL;
            v[i].head_left = v[i].head_right = NULL;
        }
        return v;
    }

    ..... Free pair in block
    .....
    void free_pair(pair1, pair2, n1, n2)
    int n1, n2;
    {
        int i;
        if (pair1 == NULL) return;
        for (i = n1; i <= n2; i++) {
            free(pair1);
            free(pair2);
        }
        free(pair1);
        free(pair2);
        pair1 = pair2 = NULL;
    }

    ..... Free block_rect
    .....
    void free_block_rect(struct block_rect *block)
    {
        free(pair(block->pairright, block->pairleft, block->upper_y, block->lower_y, block->lower_y));
        block = NULL;
    }

    ..... Free white
    .....
    void free_white(struct white *white)
    {
        free(pair(white->pairright, white->pairleft, white->upper_y, white->lower_y));
        white = NULL;
    }

    ..... Allocate memory for data_outline
    .....
    ..... Adding one element to data_outline on right side
    .....

```

```

.....
void add_x_right(x, y, ylist)
{
    struct data_outline *tmp;

    tmp = new_data_outline(y);
    if (ylist[y].total_right == 0) {
        ylist[y].total_right = tmp;
        ylist[y].headright = tmp;
    } else {
        ylist[y].current_right->next = tmp;
        ylist[y].current_right = tmp;
        ylist[y].total_right++;
    }
}

.....
void add_x_left(x, y, ylist)
{
    struct data_outline *ylist;

    struct data_outline *tmp;

    tmp = new_data_outline(x);
    if (ylist[y].total_left == 0) {
        ylist[y].total_left = tmp;
        ylist[y].headleft = tmp;
    } else {
        ylist[y].current_left->prev = tmp;
        ylist[y].current_left = tmp;
        ylist[y].total_left++;
    }
}

.....
void add_y_left(x, y, location, ylist)
{
    int n;
    struct data_outline *ylist;

    if (location == LEFT)
        add_x_left(x, y, ylist);
    else
        add_x_right(x, y, ylist);
    else if (location == R0)
        add_x_left(x, y, ylist);
    else
        add_x_right(x, y, ylist);
}

.....
void add_y_right(x, y, location, ylist)
{
    int n;
    struct data_outline *ylist;

    if (location == LEFT)
        add_x_left(x, y, ylist);
    else
        add_x_right(x, y, ylist);
    else if (location == R0)
        add_x_left(x, y, ylist);
    else
        add_x_right(x, y, ylist);
}

.....
void bubble_sort(ylist, nl, nb)
{
    int nl, nb;
}

```



—271—

[illegible]

【図70】

```

        wblock->previous->next = wblock->next;
        wblock->next->previous = wblock->previous;
    }

    /*..... remove task block .....*/
    /*..... remove task block .....*/
    void remove_taskblock(first, head)
    struct block *first, *head;
    {
        if (head->firstsubblock == first) {
            head->firstsubblock = first->next;
            if (first->next != NULL) first->next->previous = NULL;
            else head->currentsubblock = NULL;
        }
        else if (head->currentsubblock == first) {
            head->currentsubblock = first->previous;
            if (first->previous != NULL) first->previous->next = first->next;
            else { first->previous->next = first->next;
                    first->next->previous = first->previous;
                }
        }
    }

    /*..... remove nontask block .....*/
    /*..... remove nontask block .....*/
    void remove_nontaskblock(first, head)
    struct block *first, *head;
    {
        if (head->firstnontaskblock == first) {
            head->firstnontaskblock = first->next;
            if (first->next != NULL) first->next->previous = NULL;
            else head->currentnontaskblock = NULL;
        }
        else if (head->currentnontaskblock == first) {
            head->currentnontaskblock = first->previous;
            if (first->previous != NULL) first->previous->next = first->next;
            else { first->previous->next = first->next;
                    first->next->previous = first->previous;
                }
        }
    }

    /*..... Transfer block from one head to another head block .....*/
    /*..... Transfer block from one head to another head block .....*/
    void transfer_block(frontend, tostart, first, originalhead, head)
    int frontend, tostart;
    struct block *first, *originalhead, *head;
    {
        /*print("get into transfer block\n"); */
        if (frontend == tostart)
            remove_taskblock(first, originalhead);
        else
            remove_nontaskblock(first, originalhead);
        add_taskblock(tostart, first, head);
        if (originalhead != head)
            originalhead->nextsubblock = NULL;
    }
}

```



—274—





—276—

```

tapwhite->index = ablock->group;
tapwhite = tapwhite->next;
} else {
    tap = tapwhite->next;
    tapwhite->next = tapwhite->next->next;
    tapwhite = tap;
}
}

for (tap=>ablock->firstsubblock; tap != NULL; tap = tap->next)
    printf("Index = %d, subgroup = %d\n", tap->index, tap->ingroup);
}

/*----- Search block in the range specified -----*/
void block_in_range(p, x1, y1, x2, y2, headblock)
unsigned char *p, *y1, *x2, *y2;
struct block **headblocks;
{
    int struct_block_ref_n;
    struct block_ref n;
    for ((i=y1); i <= y2; i++)
        for ((j=x1); j <= x2; j++) {
            if (getch == 0) {
                if (p[i][j] == LEFTLIGHT_ONE ||
                    p[i][j] == RIGHTLIGHT_ONE)
                    continue;
            }
            if (p[i][j] == ONE) {
                n = new_struct_block(headblock);
                n->index = headblock->numsubblock;
                search_sub(p, j, x1, y1, x2, y2, n);
                printf("Index = %d len = %d wid = %d x1 = %d y1 = %d i = %d j = %d\n",
                    n->index, n->length, n->width, n->left_x, n->upper_y, i, j);
                for ((i=j <= x2); i <= x2; i++) {
                    continue;
                }
            }
            if (p[i][j] == RIGHTLIGHT_ONE)
                getch = 1;
            continue;
        }
    } else {
        if (p[i][j] == ZERO || p[i][j] == ONE) {
            if (p[i][j] == LEFTLIGHT_ONE || p[i][j] == RIGHTLIGHT_ONE)
                continue;
            if (p[i][j]) == LEFT_ZERO1
                getch = 0;
            continue;
        }
    }
}

if (block_no == 0) {
    for (i = tapwhite->upper_y; i < tapwhite->lower_y; i++)
        if (tapwhite->spareleft[i][0] != 1)
            tapwhite->line_with_black++;
}

keep = 1;
if (block_no == 0) { tapwhite->line_with_black = 0.5 * POINTS / SCALE_FACTOR; }
if (block_no == 0)
    keep = 0;
printf("Get out of block in white keep = %d\n", keep);
return keep;
}

/*----- search block in block -----*/
void search_block_in_block(nblock, p)
struct block_ref *nblock;
unsigned char *p;
{
    struct val_block *tapwhite, *tap;
    int i, block_no, keep;
    struct block_ref *tapi;
    nblock->group = 0;
    for (tapwhite = nblock->firstsubblock; tapwhite != NULL; )
        nblock->group = 0;
        if (tapwhite->index <= 4096 * SUBSUB + 4096 * GROUP + 4096 * SUBGROUP)
            printf("Tapwhite index = %d numsub = %d\n", tapwhite->index, tapwhite->numsubblock);
        keep = 0;
        if (tapwhite->numsubblock != 0)
            for (i = 0; i < tapwhite->numsubblock; i++) {
                printf("%d %d\n", i);
                if (block_in_val_block(nblock, p, n(tapwhite->subblock(i))) {
                    keep = 1;
                    if (tapwhite->subblock(i).line_with_black
                        tapwhite->line_with_black
                        tapwhite->subblock(i).line_with_black
                    )
                }
            }
            if (keep == 0) {
                for (i = 0; i < tapwhite->numsubblock; i++) {
                    free_pair(tapwhite->subblock(i));
                    tapwhite->subblock(i) = NULL;
                    if (tapwhite->subblock(i).upper_y
                        tapwhite->subblock(i).lower_y
                    )
                }
                tapwhite->spareleft = tapwhite->spareleft == NULL;
                free(struct val_block *) tapwhite->subblock(i);
            }
        } else if (block_in_val_block(nblock, p, tapwhite))
            keep = 1;
        if (keep) {

```

[圖 7 5]

```

..... count continuous ones and zero
.....
/* print("one out of block in range\n"); */
.....
void sort_block_rect(nblock, p, avg_one, std_one, avg_zero, std_zero)
.....
{
    unsigned char *p;
    float *avg_one, *std_one, *avg_zero, *std_zero;
    int i, j, k;
    int one_count, zero_count;
    int one, zero;
    int no_one, no_zero;
    /*
    print("place 3\n");
    one = vector(0, nblock->length*nblock->width-1);
    zero = vector(0, nblock->length*nblock->width-1);
    no_one = 0;
    no_zero = 0;
    for (i = nblock->upper_y; i <= nblock->lower_y; i++) {
        one_count = 0;
        zero_count = 0;
        for (j = 1; j <= nblock->pairing[1][0]; j++) {
            for (k = nblock->pairing[1][1]; k <= nblock->pairing[1][2]; k++) {
                if (p[i*nblock->width+k]) {
                    one_count++;
                    if (zero_count != 0) {
                        zero(no_zero++) = zero_count;
                        zero_count = 0;
                    }
                } else {
                    zero_count++;
                    if (one_count != 0) {
                        one(no_one++) = one_count;
                        one_count = 0;
                    }
                }
            }
        }
    }
    print("no_one = %d no_zero = %d\n", no_one, no_zero);
    avg_one = avg_zero = std_one = std_zero = 0;
    if (no_one > 0) avg_one = no_one, avg_one = std_one;
    if (no_zero > 0) avg_zero = no_zero, avg_zero = std_zero;
    print("avg_one = %f avg_zero = %f\n", avg_one, avg_zero);
    */
    free_vector(one, 0, nblock->length*nblock->width-1);
    free_vector(zero, 0, nblock->length*nblock->width-1);
    ..... count continuous ones and zero
    .....
}

```

```

.....
/* print("one out of block in range\n"); */
.....
void sort_block_rect(nblock, p, avg_one, std_one, avg_zero, std_zero)
.....
{
    unsigned char *p;
    float *avg_one, *std_one, *avg_zero, *std_zero;
    int i, j, k;
    int one_count, zero_count;
    int one, zero;
    int no_one, no_zero;
    /*
    print("place 3\n");
    one = vector(0, nblock->length*nblock->width-1);
    zero = vector(0, nblock->length*nblock->width-1);
    no_one = 0;
    no_zero = 0;
    for (i = nblock->upper_y; i <= nblock->lower_y; i++) {
        one_count = 0;
        zero_count = 0;
        for (j = 1; j <= nblock->pairing[1][0]; j++) {
            for (k = nblock->pairing[1][1]; k <= nblock->pairing[1][2]; k++) {
                if (p[i*nblock->width+k]) {
                    one_count++;
                    if (zero_count != 0) {
                        zero(no_zero++) = zero_count;
                        zero_count = 0;
                    }
                } else {
                    zero_count++;
                    if (one_count != 0) {
                        one(no_one++) = one_count;
                        one_count = 0;
                    }
                }
            }
        }
    }
    print("no_one = %d no_zero = %d\n", no_one, no_zero);
    avg_one = avg_zero = std_one = std_zero = 0;
    if (no_one > 0) avg_one = no_one, avg_one = std_one;
    if (no_zero > 0) avg_zero = no_zero, avg_zero = std_zero;
    print("avg_one = %f avg_zero = %f\n", avg_one, avg_zero);
    */
    free_vector(one, 0, nblock->length*nblock->width-1);
    free_vector(zero, 0, nblock->length*nblock->width-1);
    ..... count continuous ones and zero
    .....
}

```

```

.....
/* print("one out of block in range\n"); */
.....
void sort_block_rect(nblock, p, avg_one, std_one, avg_zero, std_zero)
.....
{
    unsigned char *p;
    float *avg_one, *std_one, *avg_zero, *std_zero;
    int i, j, k;
    int one_count, zero_count;
    int one, zero;
    int no_one, no_zero;
    /*
    print("place 3\n");
    one = vector(0, nblock->length*nblock->width-1);
    zero = vector(0, nblock->length*nblock->width-1);
    no_one = 0;
    no_zero = 0;
    for (i = nblock->upper_y; i <= nblock->lower_y; i++) {
        one_count = 0;
        zero_count = 0;
        for (j = 1; j <= nblock->pairing[1][0]; j++) {
            for (k = nblock->pairing[1][1]; k <= nblock->pairing[1][2]; k++) {
                if (p[i*nblock->width+k]) {
                    one_count++;
                    if (zero_count != 0) {
                        zero(no_zero++) = zero_count;
                        zero_count = 0;
                    }
                } else {
                    zero_count++;
                    if (one_count != 0) {
                        one(no_one++) = one_count;
                        one_count = 0;
                    }
                }
            }
        }
    }
    print("no_one = %d no_zero = %d\n", no_one, no_zero);
    avg_one = avg_zero = std_one = std_zero = 0;
    if (no_one > 0) avg_one = no_one, avg_one = std_one;
    if (no_zero > 0) avg_zero = no_zero, avg_zero = std_zero;
    print("avg_one = %f avg_zero = %f\n", avg_one, avg_zero);
    */
    free_vector(one, 0, nblock->length*nblock->width-1);
    free_vector(zero, 0, nblock->length*nblock->width-1);
    ..... count continuous ones and zero
    .....
}

```

—278—

```

void cont_grow_and_resize(nblock, p, avg_one, std_one, avg_zero, std_zero,
struct block_rect *block,
double *avg_one, *std_one, *avg_zero, *std_zero,
int no_ones, no_zeros)
{
    int i, j, x;
    int one_count, zero_count;
    int no_ones, no_zeros;

    print("place av's");
    one = vector(0, nblock->length*nblock->width-1);
    zero = vector(0, nblock->length*nblock->width-1);
    no_ones = 0; no_zeros = 0;
    for (i = 0; i < nblock->height; i++) {
        one_count = 0; zero_count = 0;
        for (j = 1; j < nblock->width; j++) {
            if (p[i][j] <= 0) {
                one_count++;
            } else {
                zero_count++;
            }
        }
        avg_one[i] = (double)one_count / (nblock->width);
        std_one[i] = sqrt((double)one_count / (nblock->width));
        avg_zero[i] = (double)zero_count / (nblock->width);
        std_zero[i] = sqrt((double)zero_count / (nblock->width));
    }
    print("avg's");
    avg_one = vector(0, nblock->length-1);
    std_one = vector(0, nblock->length-1);
    avg_zero = vector(0, nblock->length-1);
    std_zero = vector(0, nblock->length-1);
    for (i = 0; i < nblock->length; i++) {
        one_count = 0; zero_count = 0;
        for (j = 0; j < nblock->height; j++) {
            one_count += one[j][i];
            zero_count += zero[j][i];
        }
        avg_one[i] = (double)one_count / nblock->height;
        std_one[i] = sqrt((double)one_count / nblock->height);
        avg_zero[i] = (double)zero_count / nblock->height;
        std_zero[i] = sqrt((double)zero_count / nblock->height);
    }
    print("sort block");
    sort_block(nblock, p, avg_one, std_one, avg_zero, std_zero);
}

```

—279—

[illegible]

—280—

[illegible]



【图 80】

```

print("Index = %d yd used %d m = %f temp=length = %d m",
      temp_index, temp_upper_y, temp_lower_y, m, temp_length);
//
split = 0;
if (((float)temp_length > 1e6) || ((float)temp_length > 1e7)) {
    upper = (temp_upper_y-1)*redmns_factor;
    lower = temp_lower_y*redmns_factor-1;
    left = (temp_lower_y-1)*redmns_factor;
    right = temp_right_y*redmns_factor-1;
    print("place %d",
          total,
          for (i = upper; i <= lower; i++) {
              for (j = 1; j <= right; j++)
                  for (k = 0; k <= total; k++) total++;
          }
          for (i = upper; total(i) <= upper; i++)
              for (j = 1; j <= lower; j++)
                  for (k = 0; k <= total; k++) total++;
          for (i = 1; i <= total; i++)
              for (j = 1; j <= lower; j++)
                  for (k = 0; k <= total; k++) total++;
          if (total <= lower) break;
          if (total <= lower) break;
          right = (temp_upper_y-1)*redmns_factor-1;
          left = (temp_lower_y-1)*redmns_factor-1;
          if (total <= upper)
              for (i = 1; i <= temp_upper_y; i++)
                  for (j = 1; j <= temp_lower_y; j++)
                      for (k = 0; k <= temp_lower_y-1; k++)
                          total++;
          }
    print("gap 1 = %d m = %d y = %d m", gap1, gap2, temp->rc_x,
          temp_upper_y);
    //
    if (temp->firstsublock != NULL)
        block_split(headblock, temp, gap1, gap2);
    else
        block_split(headblock, temp, gap1, gap2);
    split = 1;
    break;
}
}
free_vector(total, upper, lower);
return split;
}
print("get out of test_split m");
//
second time test discrimination
//
void a_text_discriminate(headblock)
{
    int num = 0;
    float temp_length = 0;
    struct block_test *temp, *tempb;
    print("get into a_text_discriminate m");
    //
    temp_length = headblock->avg_weight;
    if (temp_factor > 1)
        for (temp = headblock->firstsublock; temp != NULL; temp = temp->next)
            if (temp->avg_weight > temp_length)
                temp_length = temp->avg_weight;
    for (temp = headblock->firstsublock; temp != NULL; temp = temp->next)
        if (temp->power == ON) {
            transfer_block(temp, tempb,
                          temp_length,
                          temp, temp,
                          temp, temp);
        }
    temp = temp->next;
    transfer_block(temp, tempb,
                  temp_length,
                  temp, headblock, headblock);
    } else {
        temp_length = (long)temp_length;
        num++;
        temp->avg_weight = temp;
        temp = temp->next;
    }
    if (num > 0)
        headblock->avg_weight = (float)(temp_length/(float)num);
    else
        headblock->avg_weight = 0.0;
    if (total_text < num) {
        total_text = num;
        headblock->avg_weight;
    }
    print("num = %d temp_length = %f m", num, temp_length);
    //
    print("get out of a_text_discriminate m");
    //
    discriminate temp and constant
    //
    void test_discriminate(headblock)
    struct block_test *headblock;
    struct block_test *temp;
    print("get into test_discriminate");
    //
    if (test_discriminate(headblock))
        a_text_discriminate(headblock);
    else {
        for (temp = headblock->firstsubblock; temp != NULL; temp = temp->next)
            test_discriminate(temp);
    }
    //
    Count the density inside the block
    //
}

```







```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

—286—

[illegible]

【图 8 5】

[illegible]

【图 8-7】

[illegible]



—290—

—290—



【图 90】

[illegible]

【图 9 1】

[illegible]

—293—

[illegible]

【図93】

```

if ((subapline->right_x < tapline->left_x) ||
    (subapline->right_x == tapline->left_x && width < w))
{
    (subapline->left_x = tapline->left_x);
    (subapline->left_y = tapline->left_y);
    (subapline->right_x = subapline->left_x + width);
    (subapline->right_y = subapline->left_y + height);
    (subapline->width = width);
    (subapline->height = height);
    (subapline->index = tapline->index);
    if (name != subapline->name, tapline->index)
    {
        if (subapline->index, tapline->index)
        {
            subapline->index = tapline->index;
            subapline->name = tapline->name;
            continue;
        }
        subapline->name = subapline->prevname;
    }
    for (tapline = firsttapline, i = 1; tapline != NULL;
        tapline = tapline->next, i++)
    {
        subapline->index = i;
    }
}

/*
    ..... Formation of line block
    .....
    void line_block(headlineblock, firstblock, column)
    struct lineblock *headlineblock;
    struct block_area *firstblock;
    int column;

    int i, j, steps, sort_index, num;
    struct lineblock *widthfill;
    struct block_rect *firstblock;
    struct block_rect *sortedblock;
    struct lineblock *taplineblock;
    fillcondition fill;

    total_group = 0; groupover = 0; groupover = 0;
    upper_row = firstblock->upper_y;
    lower_row = firstblock->lower_y;
    upper_row = (upper_row - 1) < 0 ? 0 : upper_row;
    lower_row = (lower_row + 1) > 0 ? lower_row : 0;
    if (lower_row < 0 || lower_row > 0)
    {
        numlineblock = 0;
        text_len = firstblock->avg_height * 0.5;
        firstblock->avg_height = 0.5;
        width = (int)((float)text_len * width * 0.5);
        widthfill = new_blockpool(0, column);
        for (i = 0; i < column; widthfill(i) = NULL;
            sorted_block = sort_block(text, firstblock);
            text_row = block_rect->upper_y;
            firstblock->upper_y;
            firstblock->right_x;
}
for (sort_index = 0; sort_index < num; i++)
{
    fill(filltest, sorted_block(sort_index), firstblock, widthfill);
    // group a new line //
    if (fill == NEXT)
    {
        taplineblock = new_line(headlineblock);
        taplineblock->index = numlineblock;
        group_connected_block(steps, column, taplineblock, firstblock,
            widthfill);
        continue;
    }
    else if (fill == STOP)
    {
        convert_block(widthfill(steps));
        transfer_block(text, TEXT, sorted_block(sort_index));
        firstblock, widthfill(stop);
    }
    else
    {
        for (j = 0; j < sorted_block(sort_index)->right_x; j++)
        {
            widthfill(j) = sorted_block(sort_index);
        }
        sort_index++;
    }
}

for (i = 0; i < column; i++)
{
    if (widthfill(i) != NULL)
    {
        taplineblock = new_line(headlineblock);
        steps = 1;
        sort_index = numlineblock;
        group_connected_block(steps, column, taplineblock, firstblock,
            widthfill);
        i = taplineblock->right_x;
    }
}

combine_full_overlap_line(headlineblock, firstblock);
print("..... after first combine .....");
combine_full_overlap_line(headlineblock, firstblock);
free_matrix(text_area, firstblock->upper_y, firstblock->lower_y,
    firstblock->right_x);
free(struct_block_text); widthfill;
free(struct_block_text); sorted_block;
}

//..... group line for the block contains more than one group area .....
void line_group(filllineblock, firstblock)
{
    struct lineblock *filllineblock;
    struct block_rect *firstblock;
    struct block_rect *firstblock;

    int i, m, y, j, center_y;
    struct lineblock *tapline;
    struct block_rect *tapline;
    struct block_rect *tapline;
    struct block_rect *tapline;

    tapline = new_linepoint(i, firstblock->group);
    for (i = 0; i < firstblock->group; i++)
    {
        tapline(i) = new_linepoint(i, firstblock);
    }
    for (tapline = firstblock->firstlineblock, taplineblock = NULL;
        firstblock->firstlineblock, taplineblock = taplineblock->next;
        taplineblock = nextblock;
}

```

【図94】

```

for (tapwhite = firstblock->firstwhite; tapwhite != NULL; tapwhite = tapwhite->next)
    print("white index = %d x1 = %d y1 = %d x2 = %d y2 = %d\n",
        tapwhite->index, tapwhite->left_x, tapwhite->upper_y,
        tapwhite->right_x, tapwhite->lower_y);

for (tapwhite = firstblock->firstwhite; tapwhite != NULL; tapwhite = tapwhite->next)
{
    if (tapline[0]->firstblock == NULL)
    {
        tapline[0]->firstblock = tapwhite->firstblock->firstblock;
        tapline[0]->length = tapwhite->firstblock->length;
        tapline[0]->upper_y = tapwhite->upper_y;
        tapline[0]->lower_y = tapwhite->lower_y;
    }

    if (((tapline[0]->lower_y-tapwhite->upper_y) < 0) || ((tapwhite->upper_y-tapline[0]->lower_y) < 0))
    {
        for (y = tapline[0]->upper_y-1; y > 1; y--)
        {
            if (original_p[y][x] != 0)
            {
                if (original_p[y][x] != 0) break;
            }
            if (tapline[0]->upper_y == y) tapline[0]->upper_y = y-1;
            for (y = tapline[0]->lower_y; y < 1; y--)
            {
                if (original_p[y][x] != 0)
                {
                    if (original_p[y][x] != 0) break;
                }
                if (tapline[0]->lower_y == y) tapline[0]->lower_y = y+1;
            }
        }
        for (y = tapline[0]->upper_y; y < tapline[0]->lower_y; y++)
        {
            if (original_p[y][x] != 0)
            {
                if (original_p[y][x] != 0) break;
            }
            if (tapline[0]->upper_y == y) tapline[0]->upper_y = y+1;
            if (tapline[0]->lower_y == y) tapline[0]->lower_y = y-1;
        }
    }

    for (x = tapline[0]->left_x; x < tapwhite->right_x; x++)
    {
        for (y = tapline[0]->upper_y; y < tapline[0]->lower_y; y++)
        {
            if (original_p[y][x] != 0)
            {
                if (original_p[y][x] != 0) break;
            }
            if (tapline[0]->left_x == x)
            {
                if (tapline[0]->left_x == x) break;
            }
            if (tapline[0]->right_x == x)
            {
                if (tapline[0]->right_x == x) break;
            }
        }
    }

    tapline[0]->length = tapline[0]->length;
    tapline[0]->width = tapline[0]->width;
    tapline[0]->upper_y = tapline[0]->upper_y;
    tapline[0]->lower_y = tapline[0]->lower_y;
    tapline[0]->right_x = tapline[0]->right_x;
    tapline[0]->left_x = tapline[0]->left_x;
    tapline[0]->next = tapline[0]->next;
    tapline[0]->previous = tapline[0]->previous;
    tapline[0]->index = 0;
}

tapline[0]->next = tapline[0];
tapline[0]->previous = tapline[0];
for (tap = firstblock->firstwhite; tap != NULL; tap = tap->next)
{
    if (tap->next != NULL)
    {
        tap->next->previous = tap;
        tap->previous->next = tap;
    }
}

```

【图95】

```

print("make up on id x = %d y = %d\n", tap->index, tap->left_x, tap->supper_y);
x = tap->left_x-1;
count = 0;
for (x = tap->left_x-1; x >= max((tap->left_x)-(int)text_length,
    (firstblock->left_x)); x--) {
    for (y = tap->supper_y; y <= tap->lower_y; y++)
        if (original[y][x]) count++;
    print("count = %d x = %d y1 = %d y2 = %d\n", count, x, tap->supper_y, tap->lower_y);
    if (count <= 0) break;
    for (y = tap->supper_y; y <= tap->lower_y; y++)
        for (x = tap->left_x-1; x <= firstblock->lower_x; x++)
            if (original[y][x]) break;
    else original[y][x] = 1;
    x = tap->right_x-1;
    for (x = tap->right_x-1; x <= min((tap->right_x)-(int)text_length,
        (firstblock->right_x)); x++)
        count = 0;
    for (y = tap->supper_y; y <= tap->lower_y; y++)
        for (x = tap->left_x-1; x <= firstblock->lower_x; x++)
            if (original[y][x]) count++;
    print("count = %d x = %d y1 = %d y2 = %d\n", count, x, tap->supper_y, tap->lower_y);
    if (count <= 0) break;
    for (y = tap->supper_y; y <= tap->lower_y; y++)
        for (x = tap->left_x-1; x <= firstblock->lower_x; x++)
            if (original[y][x]) break;
    else original[y][x] = 1;
}
}
}

/* ..... Initialise the formation of line block ..... */
void init_line_block(firstblock, firstblock, column, orig_row, o_p)
struct line_block *firstblock;
struct line_block *firstblock;
int column, orig_row;
unsigned char **o_p;
{
    int num;
    struct line_block *tap, *tapb;
    struct line_block *tapline;

    print("get into line\n");
    orig_row = orig_row;
    firstblock->first_block = firstblock->firstblock->first_block;
}

```

```

tap->gap((firstblock);
if (firstblock->group > 1)
    tap->group = firstblock->group;
if (firstblock->settable)
    line_group((firstblock, firstblock);
else if (firstblock->settable < 0)
    line_group((firstblock, firstblock);
else if (firstblock->first_block == NULL)
    line_block((firstblock, firstblock, column);
else if (firstblock->first_block)
    line_block((firstblock, firstblock, column);

print("get into line 1\n");
for (num=0; tap->firstblock->firstblock; tap /= NULL; 1)
    tapline = new_line_block(firstblock);
    tapline->left_x = tap->left_x;
    tapline->right_x = tap->right_x;
    tapline->supper_y = tap->supper_y;
    tapline->lower_y = tap->lower_y;
    tapline->width = tap->width;
    tapline->length = tap->length;
    tapline->block = tap->block;
    tapb = tap->next;
    remove_nextblock(tapb, firstblock);
    tap->next = NULL;
    tap->previous = NULL;
    if (((tap->settable < 0) || (tap->settable < 0))
        || ((tap->settable < 0) || (tap->settable < 0)))
        init_line_block(tapline, tapb, column, orig_row, o_p);
    else {
        tapline->first_block = tapb;
        tapline->current_block = tapb;
    }
    tapb = tapb;
}

/* ..... Print the line information into a file ..... */
void print_line_block, (firstblock)
int level;
{
    struct line_block *firstblock;
    int numlevel, i;
    struct line_block *tapubline, *tapubline;
    numlevel = level-1;
    for (tapubline = firstblock->firstblock; tapubline != NULL;
        tapubline = tapubline->next) {
        for (i = 0; i <= (level-1); i++)
            print("line %d, id %d, index = %d, tapubline, tapubline->index);
            print("line %d, id %d, index = %d, tapubline, tapubline->index);
            tapubline->left_x, tapubline->supper_y,
            tapubline->length, tapubline->width,
            tapubline->current_block, tapubline->next,
            tapubline->number, tapubline->index);
    }
    for (tapline = firstblock->firstblock; tapline != NULL;
        tapline = tapline->next) {
        for (i = 0; i <= (level-1); i++)
            print("line %d, id %d, index = %d, tapline, tapline->index);
            print("line %d, id %d, index = %d, tapline, tapline->index);
            tapline->left_x, tapline->supper_y,
            tapline->length, tapline->width,
            tapline->current_block, tapline->next,
            tapline->number, tapline->index);
    }
}

```

【図96】

```

                                tapline=tapline->next){
    for ( i = 0; i <= (level-1)*8; i++){
        fprintf(line_file, " ");
        fprintf(line_file, "%d nontext index = %d ", tapline, tapline->index);
        fprintf(line_file,
            "x%d y%d l%d w%d att = %x block index%d indexl%d g = %d l = %d r = %d\n",
            tapline->left_x, tapline->upper_y, tapline->length,
            tapline->width, tapline->att, tapline->first_block->index,
            tapline->current_block->index, tapline->group, tapline->l_caption,
            tapline->r_caption);
        print_ln(nextlevel, tapline);
    }

    /*----- Open the file for the output of line message -----*/
    /*----- Print the line information into a file -----*/

void print_lineblock(firstline)
struct lineblock *firstline;
{
    line_file = fopen("line_file.dat", "w");
    print_ln(1, firstline);
    fclose(line_file);
}

```

【図118】

```

                                tapsection = tapsection->next){
    if ((tapsection->att & (TITLE|NSOLID|VSOLID|PICTURE)) continue;
    else
        ini_section_block(tapsection, tapsection->firstlineblock, p,
                           column);

    /*----- Print the line information into a file -----*/

void print_sec(level, firstsec)
int level;
struct sectionblock *firstsec;
{
    int nextlevel, i;
    struct sectionblock *tapsubsec, *tapsec;
    nextlevel = level+1;
    for (tapsubsec = firstsec->firstsubsec; tapsubsec != NULL;
        tapsubsec = tapsubsec->next){
        for ( i = 0; i <= (level-1)*8; i++){
            printf(" ");
            printf("index = %d ", tapsubsec->index);
            printf("x%d y%d l%d w%d att = %x\n",
                tapsubsec->left_x, tapsubsec->upper_y,
                tapsubsec->length, tapsubsec->width, tapsubsec->att);
        }

        for (tapsec = firstsec->firstnontextsection; tapsec != NULL;
            tapsec = tapsec->next){
            for ( i = 0; i <= (level-1)*8; i++){
                printf(" ");
                printf("nontext index = %d ", tapsec->index);
                printf("x%d y%d l%d w%d att = %x\n",
                    tapsec->left_x, tapsec->upper_y, tapsec->length,
                    tapsec->width, tapsec->att);
                print_sec(nextlevel, tapsec);
            }
        }

        /*----- Open the file for the output of line message -----*/
        /*----- Print the section information into a file -----*/

void print_secblock(firstsec)
struct sectionblock *firstsec;
{
    print_sec(1, firstsec);
}

```

【図217】

## Source Code for Mono-Spaced ("Courier") Segmentation

【图 9-7】

```

for ( i = n1; i <= n2; i++)
    v[i].column_no = 0;
return v;
}

..... allocation sectionblockpointer
.....
struct sectionblock **v;
{
    int n1, n2;

    v = (struct sectionblock **) malloc((n2-n1+1)*sizeof(struct sectionblock));
    if (!v)
        printf("error: in the allocation of sectionblock\n");
    exit(1);
    v = v+n1;
}

return v;
}

..... sort line block pointer array
.....
void sort_linepointer(line, num)
struct lineblock **line;
int num;
{
    int i, j, swap;
    struct lineblock *tmp;

    for ( i = 0; i < num-1; i++) {
        for ( j = i+1; j < num; j++) {
            if (line[i]-supperly > line[j]-supperly) {
                tmp = line[i];
                line[i] = line[j];
                line[j] = tmp;
            }
        }
        if (swap == 0) break;
    }
}

..... line block sorting
.....
struct lineblock **sort_y(firstlineblock, numline;
int *numline;
{
    struct lineblock *tmp;
    struct lineblock **line_x;
    int i, j, swap;

    *numline = 0;
    for ( tmp = firstlineblock->firstlineblock; tmp != NULL; tmp = tmp->next)
        (*numline)++;
    for ( tmp = firstlineblock->firstlineblock; tmp != NULL; tmp = tmp->next)
        (*numline)++;

    line_x = new_linepointer(0, (*numline)-1);
}

```



【図98】

```

for (i = 0; tap = firstlineblock->firstsubline; tap != NULL; tap = tap->next)
    line_x[i++] = tap;
/* extra testing */
if (((firstlineblock->currentsubline != NULL) &&
    (i != firstlineblock->currentsubline->index)))
    printf("error in sort\n");
for (tap = firstlineblock->firstsubline; tap != NULL; tap = tap->next)
    line_x[i++] = tap;
sort_linepointer(line_x, "mainline");
return line_x;
}
/*..... section block setting .....*/
/*..... Chain the context section block .....*/
struct sectionblock *new_section(headsection)
struct sectionblock *headsection;
{
    struct sectionblock *n;
    n = new_sectionblock(0, 0);
    if (headsection->firstsubsection == NULL)
        headsection->firstsubsection = n;
    else
        headsection->currentsubsection->next = n;
    headsection->currentsubsection = n;
    return n;
}
/*..... Chain the section block .....*/
struct sectionblock *new_section(headsection)
struct sectionblock *headsection;
{
    struct sectionblock *n;
    n = new_sectionblock(0, 0);
    if (headsection->firstsubsection == NULL)
        headsection->firstsubsection = n;
    else
        headsection->currentsubsection->next = n;
    headsection->currentsubsection = n;
    return n;
}
/*..... Take lineblock from section .....*/
void take_section(line, currentsection)
struct lineblock *line;
struct sectionblock *currentsection;
{
    int i;
    struct sectionblock *v;
    v = (struct sectionblock *) malloc(sizeof(struct sectionblock));
    if (i % 100)
        printf("allocation error: section block %d\n", i);
    exit(1);
}

```

【図 99】

```

(
    if (current->firstlineblock == line)
        current->firstlineblock = line->next;
    if (current->currentlineblock == line)
        current->currentlineblock = NULL;
    else
        current->firstlineblock->previous = NULL;
    ) else if (current->currentlineblock == line) {
        current->currentlineblock = line->previous;
        line->previous->next = NULL;
    } else {
        line->previous->next = line->next;
        line->next->previous = line->previous;
        line->previous = NULL;
        line->next = NULL;
    }
}
/*..... put lineblock into section .....*/
/*.....*/
void put_section(line, current)
struct lineblock *line;
struct sectionblock *current;
{
    if (current->firstlineblock == NULL) {
        current->firstlineblock = line;
        line->previous = NULL;
        line->next = NULL;
    } else {
        current->currentlineblock->next = line;
        line->previous = current->currentlineblock;
        line->next = NULL;
    }
}
/*..... create new section structure .....*/
/*.....*/
struct sam_line *new_samline(struct
struct lineblock *curr;
{
    struct samline *v;
    v = (struct samline *) malloc(sizeof(struct samline));
    if (!v) { printf("allocation error: in samline\n");
               exit(1);
    }
    v->current = curr;
    v->previous = NULL;
    v->next = NULL;
    v->connection = v->connection = NULL;
    return v;
}
/*..... sort samline array by x .....*/
/*.....*/
void sort_samline(struct
struct samline *first;

```





【図 102】

[illegible]

—304—

—304—

[illegible]

[illegible]



[illegible]



【図108】

```

/***** remove current section from same head section *****/
/***** combine two sections *****/
void rm_section(head, section)
{
    struct sectionblock *head, *section;

    if (head->currentsection == section)
    {
        if (section->previous != NULL)
        {
            section->previous->next = NULL;
        }
        else
        {
            head->firstsubsection = NULL;
        }
        head->currentsection = section->previous;
    }
    else if (head->firstsubsection == section)
    {
        head->firstsubsection = section->next;
        section->next->previous = NULL;
    }
    else
    {
        section->next->previous = section->previous;
        section->previous->next = section->next;
    }
}

/***** remove section from same head section *****/
/***** combine two sections *****/
void rm_section(head, section)
{
    struct sectionblock *head, *section;

    if (head->currentsection == section)
    {
        if (section->previous != NULL)
        {
            section->previous->next = NULL;
        }
        else
        {
            head->firstsubsection = NULL;
        }
        head->currentsection = section->previous;
    }
    else if (head->firstsubsection == section)
    {
        head->firstsubsection = section->next;
        section->next->previous = NULL;
    }
    else
    {
        section->next->previous = section->previous;
        section->previous->next = section->next;
    }
}

/***** combine two sections *****/
/***** combine two sections *****/
void combine(first, second, head)
{
    struct sectionblock *first, *second, *head;

    /* print('combine id => %d %d', second->index, first->index); */
    second->firstlineblock->prevous = first->currentlineblock;
    first->currentlineblock->next = second->firstlineblock;
    first->currentlineblock = second->currentlineblock;
    if (second->upper_y > first->upper_y)
    {
        first->upper_y = second->upper_y;
    }
    if (second->lower_y > first->lower_y)
    {
        first->lower_y = second->lower_y;
    }
    if (second->left_x < first->left_x)
    {
        first->left_x = second->left_x;
    }
    if (second->right_x > first->right_x)
    {
        first->right_x = second->right_x;
    }
    first->width = first->right_x - first->left_x + 1;
    first->length = first->lower_y - first->upper_y + 1;
    if ((second->caption) != (first->caption))
    {
        rm_section(head, second);
    }
    else
    {
        rm_section(head, second);
        free(struct sectionblock *second);
    }
}

/***** Test if boundary exists between two sections *****/
/***** Test if boundary exists between two sections *****/
int boundary(first, topsec, bottomsec, align)
{
    struct sectionblock *first, *topsec, *bottomsec, *align;
    int x1, x2, y1, y2, x, y, count;
    unsigned char p;

    struct sectionblock *top;
    struct sectionblock *bottom;
    struct sectionblock *align;
    int x1, x2, y1, y2, x, y, count;
    unsigned char p;

    print('line into boundary input');
    yoverlap = cover_topsec(bottomsec->upper_y, bottomsec->lower_y,
        topsec->upper_y, topsec->lower_y);
    yoverlap = 0;
    if (topsec->caption != bottomsec->caption)
    {
        print('under caption input');
        topsec->caption = bottomsec->caption;
        topsec->firstlineblock->prevous = first->currentlineblock;
        topsec->firstlineblock->next = first->firstlineblock;
        topsec->firstlineblock->upper_y = first->upper_y;
        topsec->firstlineblock->lower_y = first->lower_y;
        if ((bottomsec->right_x <
            (topsec->firstlineblock->under_caption->left_x+text)))
        {
            topsec->firstlineblock->under_caption->right_x = text;
        }
        return 1;
    }
    return 1;
}

```

【圖 1 0 9】

```

if (yoverlap == 1.0) {
    if (align == TOPLEFT) {
        x1 = bottomsec->left_x;
        x2 = topsec->left_x;
        y1 = topsec->bottom_y;
        y2 = bottomsec->top_y;
    } else {
        x1 = topsec->right_x;
        x2 = bottomsec->right_x;
        y1 = topsec->bottom_y;
        y2 = bottomsec->top_y;
    }
} else if (align == TOPRIGHT) {
    x1 = topsec->right_x;
    x2 = bottomsec->right_x;
    y1 = topsec->bottom_y;
    y2 = bottomsec->top_y;
} else if (align == BOTTOMLEFT) {
    x1 = topsec->left_x;
    x2 = bottomsec->left_x;
    y1 = topsec->top_y;
    y2 = bottomsec->bottom_y;
} else {
    x1 = topsec->left_x;
    x2 = bottomsec->left_x;
    y1 = topsec->top_y;
    y2 = bottomsec->bottom_y;
}

count = 0;
for (s = x1; s <= x2; s++) {
    for (y = y1; y <= y2; y++) {
        if ((original[s][y] & 0x01) == 0x01) {
            count++;
        }
    }
}

if (count > 0) return 1;
if (count > 0) return 1;

printf("place 2 x1 = %d x2 = %d y1 = %d y2 = %d\n", x1, x2, y1, y2);

/*
if (x2 > x1)
for (tmp = first->firstsubsec; tmp != NULL; tmp = tmp->next) {
    if ((tmp->bottomsec) != (tmp->topsec)) continue;
    topcover = cover_range(tmp->left_x, tmp->right_x, y1, y2);
    bottomcover = cover_range(tmp->left_x, tmp->right_x, x1, x2);
    if (((topcover == 0.0) || (topcover == 1.0)) ||
        ((bottomcover == 1.0) || (bottomcover == 0.0))) continue;
    else return 1;
}

/* printf("x1 = %d x2 = %d y1 = %d y2 = %d\n", x1, x2, y1, y2);

/*
for (tmp = first->firstsubsec; tmp != NULL; tmp = tmp->next) {
    topcover = cover_range(tmp->left_x, tmp->right_x,
        tmp->left_x, tmp->right_x);
    bottomcover = cover_range(tmp->left_x, tmp->right_x,
        tmp->left_x, tmp->right_x);
    if (((topcover == 0.0) || (topcover == 1.0)) ||
        ((bottomcover == 0.0) || (bottomcover == 1.0)))
        continue;
    else return 1;
}

/*
if ((tmp->lower_y == topsec->upper_y) ||
    (tmp->upper_y == bottomsec->lower_y) ||
    (tmp->left_x == max(tmp->right_x, bottomsec->right_x)))

```

[illegible]



【图 1 1 2】

[illegible]

【図113】

```

.....
//..... average of the line .....
//.....
float avg_line_length(firstline)
{
    struct lineblock *firstline;

    int i;
    struct lineblock *tapline;
    float total = 0.0;

    for (tapline = firstline->firstsubline, i = 0; tapline != NULL;
        tapline = tapline->next, i++)
        total += (float) tapline->length;

    if (i > 0) total /= (float)i;
    return total;
}

//..... Attribute text .....
//.....
void attribute_text(tapmac)
{
    struct sectionblock *tapsect;
    struct lineblock *tapline;
    struct block *tapt;
    int x, y, dx, yi;

    for (tapse = tapsect->firstsubsection, tapse != NULL; tapse = tapse->next;
        for (tapline = tapse->firstlineblock, tapline != NULL; tapline = tapline->next;
            for (tapt = tapline->firstblock, tapt != NULL; tapt = tapt->next;
                for (y = tapse->supper_y; y <= tapse->lower_y; y++)
                    for (x = tapse->left_x; x <= tapse->right_x; x++)
                        original_p[y][x] = TEXTPRDEL;

            )
        )
    )
    for (tapse = tapsect->firstsubsection, tapse != NULL; tapse = tapse->next;
        if ((tapse->right_x - tapse->left_x) < TEXTPRDEL) continue;
        for (tapline = tapse->firstlineblock, tapline != NULL; tapline = tapline->next;
            for (tapt = tapline->firstblock, tapt != NULL; tapt = tapt->next;
                y)++
            )
        )
    )
    for (tapse = tapsect->firstsubsection, tapse != NULL; tapse = tapse->next;
        for (tapline = tapse->firstlineblock, tapline != NULL; tapline = tapline->next;
            for (tapt = tapline->firstblock, tapt != NULL; tapt = tapt->next;
                for (y = tapse->supper_y; y <= tapse->lower_y; y++)
                    for (x = tapse->left_x; x <= tapse->right_x; x++)
                        if (original_p[y][x] != TEXTPRDEL,
                            original_p[y][x] = TEXTPRDEL;
                        )
            )
        )
    )
    for (tapse = tapsect->firstsubsection, tapse != NULL; tapse = tapse->next;
        if ((tapse->right_x - tapse->left_x) < TEXTPRDEL) continue;
        for (tapline = tapse->firstlineblock, tapline != NULL; tapline = tapline->next;
            for (tapt = tapline->firstblock, tapt != NULL; tapt = tapt->next;
                y)++
            )
        )
    )
    for (tapse = tapsect->firstsubsection, tapse != NULL; tapse = tapse->next;
        for (tapline = tapse->firstlineblock, tapline != NULL; tapline = tapline->next;
            for (tapt = tapline->firstblock, tapt != NULL; tapt = tapt->next;
                for (y = tapse->supper_y; y <= tapse->lower_y; y++)
                    for (x = tapse->left_x; x <= tapse->right_x; x++)
                        if (original_p[y][x] != TEXTPRDEL,
                            original_p[y][x] = TEXTPRDEL;
                        )
            )
        )
    )
}

//..... inner section text and combine .....
//.....
void inner_section_text(firstsection)
{
    struct sectionblock *firstsection;
    struct sectionblock *tap1, *tap2;
}

```











[illegible]

[図120]

```

for (tmp = vtable, num = 0, top = NULL, top = top->next, num++)
    top[num] = tmp;

for (i = 0, j = num; i++) {
    for (k = 0, stop = 0; k < num - j - 1; k++)
        if (top[k] != white->mod_right) top[k] = white->mod_right;
    top = top[j+1];
    top[j] = top;
    stop = 1;

    if (stop == 0) break;

    top2 = -1;
    for (i = 0, j = num; i++) {
        if (top[i] != white->mod_right) top[i] = white->mod_right + OFFST;
        if (top[i] != white->mod_right + top)
            top2 = top[i] - white->mod_right;
        } else {
            top2 = top[i] - white->mod_right;
            break;
        }
    }

    if (top2 == -1) top2 = top;
}

/*..... (list top top_lowery) .....*/
void first_lowcopy (vtable, top1, top2)
struct white_for_table *vtable;
int *top1, *top2;
{
    struct white_for_table *tmpw, *top;
    int num, stop, i, j;

    for (tmp = vtable, num = 0, top1 = NULL, top = top->next, num++)
        topw = (struct white_for_table *) malloc(sizeof(white_for_table));
    for (tmp = vtable, num = 0; tmp != NULL; tmp = tmp->next, num++)
        topw[num] = tmp;

    for (i = 0, j = num; i++) {
        for (k = 0, stop = 0; k < num - i - 1; k++)
            if (topw[k] != white->mod_lowery) topw[k] = white->mod_lowery;
        topw[j+1] = topw[i];
        topw[i] = topw;
        stop = 1;

        if (stop == 0) break;

        top3 = -1;
        for (l = 0, m = num; l++) {
            for (n = 0, stop = 0; n < num - l - 1; n++)
                if (topw[n] != white->mod_lowery) topw[n] = white->mod_lowery;
            topw[m+1] = topw[l];
            topw[l] = topw;
            stop = 1;

            if (stop == 0) break;

            top4 = -1;
            for (o = 0, p = num; o++) {
                for (q = 0, stop = 0; q < num - o - 1; q++)
                    if (topw[q] != white->mod_lowery) topw[q] = white->mod_lowery + OFFST;
                if (topw[o] != white->mod_lowery + top)
                    top4 = topw[o] - white->mod_lowery;
            } else {
                top4 = topw[o] - white->mod_lowery;
                break;
            }
        }
    }
}

```







[illegible]

4

```

inc !cable(nblock)
struct block_rect nblock;

```

【図125】

```

/*****
***** File: block_util.c
***** Author: Shin-Yuan Wang
***** Date : 1-28-92
***** Copyright 1992 Canon Information Systems
*****/

/***** test if covered range overlap *****/

int if_overlap(a1, a2, b1, b2)
{
    int a1, a2, b1, b2;
    if ((a2 < b1) || (b2 < a1)) return 0;
    return 1;
}

/***** minimum of two integers *****/

int min(a, b)
{
    int a, b;
    if (a <= b) return a;
    else return b;
}

/***** ratio of overlappint *****/

float cover_range(a1, a2, b1, b2)
{
    int a1, a2, b1, b2;
    float range = 0.0;
    int x1, x2;
    if ((a2 < b1) || (b2 < a1)) return range;
    else {
        x1 = max(a1, b1);
        x2 = min(a2, b2);
    }
    range = (float)(x2-x1+1)/(float)(a2-a1+1);
    return range;
}

```

【図131】

```

/*****
***** Filename: blockline.h
***** Author: Shin-Yuan Wang
***** Date: 1-15-92
***** Copyright 1992 Canon Information Systems
*****/

#define EMPTY 0
#define OVERLAP 1
#define NEXT 2
#define SURROUNDIN 3
#define SURROUNDOUT 4

#define TH 1.7
#define WIDTHSTD 1.2
#define CAPLENGTH 0

#define DOT_BOTTOM 0.3
#define DOT_WIDTH 2
#define LINEOVERLAP 0.5

#define LEFTCAP 0x00
#define RIGHTCAP 0x01
#define NOISE_NO 3

typedef int fillcondition;

extern void as_draw_box();
extern void convert_to_headblock();
extern void transfer_block();
extern void remove_textblock();
extern int max();
extern int min();
extern float cover_range();
extern struct blockline **new_linepointer();
extern void sort_linepointer();

```

【図127】

```

/*****
*****  Filename: block_main.h
*****  Author: Shin-Yuan Wang
*****  Date: 1-11-92
*****  Copyright 1992 Canon Information Systems
*****/

#define TESTTIME 20
#define TEXT_BLOCK 0x11,
#define LINEPICTURE_BLOCK 0x21
#define MPICTURE_BLOCK 0x31
#define TABLE_BLOCK 0x41
#define LINE_BLOCK 0x51
#define FRAME_BLOCK 0x61
#define UNKNOWN_BLOCK 0x71

#define TEXT_COLOR 0
#define TABLETEXT_COLOR 1
#define LINE_COLOR 2
#define PICTURE_COLOR 3
#define MPICTURE_COLOR 4
#define UNKNOWN_COLOR 5
#define TABLE_COLOR 6
#define FRAME_COLOR 7
#define LINECAP_COLOR 8

enum {SP_READ = 0x01,
      SP_REDUCE = 0x02,
      SP_COPY = 0x04,
      SEP_BLOCK = 0x08};

struct block_image
{
    int width;
    int height;
    unsigned char** pixel;
};

int xoff[TESTTIME][2] = {(0, 1), (1, 2), (0, 1), (1, 3), (0, 1),
                          (1, 2), (0, 0), (0, 0), (1, 2), (1, 2),
                          (2, 2), (2, 2), (1, 0), (2, 1), (1, 0),
                          (3, 1), (0, 1), (0, 1), (1, 3), (1, 2)};

int yoff[TESTTIME][2] = {(0, 0), (0, 0), (1, 1), (1, 1), (2, 2),
                          (2, 2), (0, 1), (1, 2), (0, 1), (1, 2),
                          (0, 1), (0, 2), (0, 1), (0, 1), (1, 2),
                          (1, 2), (1, 2), (0, 1), (1, 2), (0, 1)};

#define TOTALCOLOR 7

char *color_label[] = {
    "Text",
    "Text in the table",
    "Picture",
    "Line Drawing Graphics",
    "Solid line (Vertical or Horizontal)",
    "Table",
    "Frame"
};

int color_code[] = {TEXT_COLOR, TABLETEXT_COLOR, MPICTURE_COLOR, PICTURE_COLOR,
                    LINE_COLOR, TABLE_COLOR, FRAME_COLOR};

```

【図132】

```

/*****
*****  Filename: blocksection.h
*****  Author: Shin-Yuan Wang
*****  Date: 1-11-92
*****  Copyright 1992 Canon Information Systems
*****/

#define GAP 15
#define OVERLAP 0.8
#define NOVERLAP 0.5
#define LEVELOVERLAP 0.0
#define LEFTSIDE 0x10
#define LEFTALIGN 0x20
#define LEFTTOWLONGER 0x40
#define LEFTBOTLONGER 0x20
#define RIGHTSIDE 0x01
#define RIGHTALIGN 0x08
#define RIGHTTOWLONGER 0x04
#define RIGHTBOTLONGER 0x01
#define TEXTPIXEL 0x01

#define NOISE_COUNT 5

extern int line_inside();
extern int max();
extern int min();
extern int if_overlap();
extern void xa_draw_box();
extern float cover_range();
void exit();

struct segment_gap
{
    int column_no;
    int *begin_column;
    int *end_column;
};

```

[illegible]

[illegible]

【図135】

```

/*****
 * Filename: Table.h
 * Author: Shin-Yuan Wang
 * Date: 1-15-93
 * Copyright 1992 Canon Information Systems
 *****/

#include "blockgeneral.h"

#define NONE 0
#define OFFSET 3
#define OFFSET2 6
#define MDX 1
#define OLD 2
#define MONSTOP 0
#define STOP 1
#define TABLE_TESTTIME 2

struct white_for_table
{
    int status;
    struct whiteblock *white;
    struct white_for_table *previous, *next;
};

```

【図136】

```

/*****
 * Filename: images.h
 * Author: Shin-Yuan Wang
 * Date: 1-15-93
 * Copyright 1992 Canon Information Systems
 *****/

#include<stdio.h>
#include<stdlib.h>

struct images
{
    int sizeL;
    int sizeC;
    int dpi;
    unsigned char **pixel;
};

```

【図142】

```

/*
 * Filename: autoinput.h
 * Header file for autoinput.c
 */

/* Function Definitions for mymalloc.c */
#ifdef CPP
/* Definition for C++ files */
extern "C" {
    void auto_initialize();
    int auto_input_int();
    void auto_terminate();
}
#else
/* Definition for C files */
void auto_initialize();
int auto_input_int();
void auto_terminate();
#endif

```

【図145】

```

/*****
 * void Zerovpp_terminate(mzlist)
 * struct NonZerovppList *mzlist; /* The list */
 *
 * {
 *     free_vector(mzlist->list); /* Deallocate memory */
 *     mzlist->allocated = 0; /* Set so won't be accidently used */
 *     mzlist->size = 0;
 * }
 *
 * void Zerovpp_add(mzlist, a, b)
 * struct NonZerovppList *mzlist; /* The list */
 * int a, b; /* Start and stop of the zerovpp */
 *
 * {
 *     if (mzlist->size+2 > mzlist->allocated) {
 *         printf("NonZerovppList FULL!\n");
 *         exit(1);
 *     }
 *     mzlist->list[mzlist->size++] = a;
 *     mzlist->list[mzlist->size++] = b;
 * }
 */

```

【図214】

```

/* If your routines are in C++ then CPP must be defined in make file */
#ifdef CPP
extern "C" void xa_set_foreground(int x, int y, int b);
extern "C" void xa_set_background(int x, int y, int b);
extern "C" void xa_init();
extern "C" void xa_flush();
extern "C" void xa_plotxy(int x, int y);
extern "C" void xa_unplotxy(int x, int y);
extern "C" void xa_redisplay();
extern "C" void xa_dia_bitmap(int x, int y, int width, int height, char *buf);
extern "C" void xa_clr_vin();
extern "C" void xa_draw_box(int x, int y, int width, int height);
extern "C" void xa_string(int x, int y, char *str);
extern "C" void xa_setcolor(int color);
extern "C" void xa_vin_dia(int width, int height);
#endif

```





【図139】

```

// File name: 3line_logic.c
// This routine contains optimized 3line logic routines
// =====
#include <images.h>
// =====
// 3x1 OR logic
void tline_OR_31(cimages oldline, cimages newline)
{
    int oldr, oldc; // size of old image
    int newr, newc; // size of new image
    unsigned char *ocr, *oc2, *oc3, *ncr; // The old & new row pointers
    unsigned char *ncr2, *ncr3; // The old & new row pointers
    oldr = oldline.alise;
    oldc = oldline.alise;
    newr = (oldr-1)/2; // Should be **
    newc = oldline.alise;
    newline.reallocate(newr, newc);
    a = oldr - newr*2; // How much is partially covered in the last row
    if (a == 3) // It's evenly aligned
        newr++;
    ocr = oldline.pixel;
    nc = newline.pixel;
    for (i=0; i<newr; i++) // Do the middle
    {
        ocr = *ocr++;
        nc = *ncr++;
        for (j=0; j<newc; j++)
            *(ncr++) = *(ocr++) | *(oc2++) | *(oc3++);
    }
    // Do the ends
    nc = *ncr;
    if (a == 1)
    {
        ocr = *ocr;
        for (j=0; j<newc; j++)
            *(ncr++) = *(ocr++) | *(oc2++) | *(oc3++);
    }
    else (
        if (a == 2) // a == 2
        {
            ocr = *ocr;
            oc2 = *ocr;
            for (j=0; j<newc; j++)
                *(ncr++) = *(ocr++) | *(oc2++) | *(oc3++);
        }
    )
}

// =====
// 1x3 AND 2x1 OR logic
void tline_AND_13_OR_21(cimages oldline, cimages newline)
{
    int oldr, oldc; // size of old image
    int newr, newc; // size of new image
    int i, j, a, b;
    unsigned char *ocr, *oc2, *ncr; // The old & new column pointers
    unsigned char *ocr2, *ocr3; // The old & new row pointers
    ocr = oldline.alise;
    oc2 = oldline.alise;
    newr = (oldr-1)/2; // Should be **
    newc = (oldc-1)/2; // Should be **
    newline.reallocate(newr, newc);
    a = oldr - newr*2; // How much is partially covered in the last row
    if (a == 3) // It's evenly aligned
        newr++;
    b = oldc - newc*2; // How much is partially covered in the last column
    if (b == 2)
    {
        newc++;
        b = 0; // == 0 for optimization of ends below
    }
    ocr = oldline.pixel;
    nc = newline.pixel;
    for (i=0; i<newr; i++) // Do the middle
    {
        ocr = *ocr;

```

【図140】

```

temp1 = *(oc1++);
*inc++ = (temp1 & *oc1++) | (temp2 & *oc1++) | (temp3 & *oc1++);
if (b) // Do the right end
    *inc = (*oc1) | (*oc2) | (*oc3);
}
// Do the bottom line (if a == 1 or 2)
nc = *nc;
if (a == 1) // a == 1 it's one row
{
    oc1 = *or;
    for (j=0; j<newc; j++)
    {
        temp1 = *(oc1++);
        *inc++ = temp1 & *oc1++;
        if (b) // Do the right end
            *inc = *oc1;
        else
            if (a == 2) // a == 2 it's two rows
            {
                oc1 = *(oc1++);
                oc2 = *or;
                for (j=0; j<newc; j++)
                {
                    temp1 = *(oc1++);
                    temp2 = *(oc2++);
                    *inc++ = (temp1 & *oc1++) | (temp2 & *oc1++);
                    if (b) // Do the right end
                        *inc = *oc1 | *oc2;
                }
            }
    }
}

oc2 = *(oc1++);
nc = *nc;
for (j=0; j<newc; j++)
{
    temp1 = *(oc1++);
    temp2 = *(oc2++);
    *inc++ = (temp1 & *oc1++) | (temp2 & *oc1++);
    if (b) // Do the right end
        *inc = (*oc1) | (*oc2);
}
// Do the bottom line (if a == 1)
// a == 1 it's one row
if (a == 1)
{
    oc1 = *or;
    for (j=0; j<newc; j++)
    {
        temp1 = *(oc1++);
        *inc++ = temp1 & *oc1++;
    }
    if (b) // Do the right end
        *inc = *oc1;
}
}

// *****
// In2 AND in1 in logic
void flip_and_in2_in1(cimages oldline, cimages newline)
{
    int oldc, oldr; // size of old image
    int newc, newr; // size of new image
    int i, j, a, b; // The old & new column pointers
    unsigned char *oc1, *oc2, *oc3, *nc; // The old & new row pointers
    register unsigned char temp1, temp2, temp3;

    oldc = oldline.sizec;
    oldr = oldline.sizer; // Should be ==
    newc = newline.sizec;
    newr = newline.sizer; // Should be ==
    newline.reallocate(newr-1, newc-1);

    a = oldr - newr; // How much is partially covered in the last row
    if (a == 1) // It's evenly aligned
        b = oldc - newc; // How much is partially covered in the last column
    if (b == 2)
    {
        newc++; // == 0 for optimization of ends below
        b = 0;
    }
    or = oldline.pixel;
    nr = newline.pixel;
    for (i=0; i<newr; i++) // Do the middle
    {
        oc1 = *(oc1++);
        oc2 = *(oc2++);
        oc3 = *(oc3++);
        nc = *(nc++);
        for (j=0; j<newc; j++)
        {
            temp1 = *(oc1++);
            temp2 = *(oc2++);

```

【図141】

```

/* filename: autolinput.h
 * This file contains the code to input from a filename
 * .....
#include <stdio.h>
#include "autolinput.h"
/* Global variables */
int auto_eof_detected;
FILE *auto_fp;
int save_count;
char filename[100];
void auto_initialize()
{
    int not_exit = 1;
    auto_eof_detected = 0;
    while(not_exit)
    {
        printf("Input filename for auto-input? ");
        scanf("%s", filename);
        auto_fp = fopen(filename, "r");
        if (auto_fp == NULL) /* Read open failed */
        {
            auto_fp = fopen(filename, "w");
            if (auto_fp == NULL) /* Write open failed */
            {
                printf("cannot open file %s\n", filename);
            }
            else /* Opened new file */
            {
                printf("Enter 1 at the next prompt.\n");
                not_exit = 1;
                auto_eof_detected = 1;
            }
        }
        else /* Read open succeeded */
        {
            printf("Opening existing file %s\n", filename);
            not_exit = 0;
        }
    }

    printf("Do you wish to save the auto-input into this file? (1 = yes) ");
    save_to_file = auto_input_int();
    printf("Manual Change the int '1' to '0' to turn off appending to the file\n");
    save_count = 0;
}

int auto_input_int()
{
    int i;
    if (auto_eof_detected)
    {
        printf("Manual Input 1-9 = newline, -9999 = exit) ? ");
        scanf("%d", &i);
        if (i == -9999)
        {
            fclose(auto_fp);
            exit(0);
        }
        if (save_to_file == 1)
        {
            if (i == -9) /* Print new lines */
            {
                printf(auto_fp, "\n"); /* Print the new lines */
                fflush(auto_fp);
                save_count = 0;
            }
            return auto_input_int(); /* And input again */
        }
        printf(auto_fp, "%d ", i);
        fflush(auto_fp);
        if (++save_count > 20)
        {
            save_count = 0;
            printf(auto_fp, "\n");
        }
        return i;
    }
    if (feof(auto_fp, "d", &i) == EOF)
    {
        printf("\n..... END OF FILE DETECTED .....");
        if (save_to_file == 1)
        {
            fclose(auto_fp);
            auto_fp = fopen(filename, "a");
            if (auto_fp == NULL)
            {
                printf("..... WARNING: CANNOT APPEND TO FILE. Is ....\n", filename);
                save_to_file = 0;
            }
            printf(auto_fp, "\n\n");
            auto_eof_detected = 1;
            return auto_input_int();
        }
        printf("\n\n");
        return i;
    }
}

printf("Do you wish to save the auto-input into this file? (1 = yes) ");
save_to_file = auto_input_int();
printf("Manual Change the int '1' to '0' to turn off appending to the file\n");
save_count = 0;
}

int auto_input_int()
{
    int i;
    if (auto_eof_detected)
    {
        printf("Manual Input 1-9 = newline, -9999 = exit) ? ");

```

【図143】

```

/* filename: charlist.c
 * This file contains Charlist and NonZeroCharlist utility routines
 * .....
 */

/* .....
 * Routines to handle Character lists
 * .....
 */
/* By Christopher Sherrick */
#include "charlist.h"
#include "gutil.h"

/* .....
 * Initialize and allocate a Charlist list up curve pointers) */
/* Call this routine once only! */
void Charlist_initialize(
    int clist_max, /* The max number of characters */
    int clist_maxh, /* Height of the line (to allocate curves) */
    int i, /* A looping var */
    /* Allocate memory for characters */
    clist_ptr = (struct Character *) malloc((unsigned) clist_max
        * sizeof(struct Character));
    if (clist_ptr == 0)
        exit(1);
    /* Memory Allocation Error in Charlist_initialize()! */
    exit(1);
    clist_maxh = 0; /* Number of characters */
    clist_max = clist_max; /* Size Allocated */
    clist_ptr = clist_ptr; /* The first character */
    clist_maxh = clist_maxh; /* vector height */

    /* Allocate memory for curves and set pointers */
    for (i = 0; i < clist_max; i++)
        clist_ptr[i].curves = vector(vector_height);
    clist_ptr[i].curves = vector(vector_height);
}

/* .....
 * Zero a previously allocated Charlist */
void Charlist_zero(clist)
    struct Charlist *clist;
{
    clist_max = 0; /* Number of characters */
    clist_ptr = 0; /* No first character */
}

/* .....
 * Remove all characters from Charlist */
void Charlist_remove(clist)
    struct Charlist *clist;
{
    int i;
    for (i = 0; i < clist_max; i++) { /* Free curves */
        free(vector(clist_ptr[i].curves));
        free(vector(clist_ptr[i].curves));
    }
}

```

```

}
free(struct Character *) clist_ptr;
clist_max = 0;
clist_ptr = 0;
clist_ptr = 0;
}

/* .....
 * Places a new character in the charlist: */
/* Takes the character beyond the charlist (.chr[alist]),
 * calculates its size, and inserts it into the list */
void Charlist_insert(struct Charlist *clist)
{
    struct Character *newchar; /* Pointer to the new character adding */
    newchar = (struct Character *) malloc(sizeof(struct Character));
    /* Calculate the position of the new character */
    Charlist_recalculate_loc(newchar);
    /* Check for full list */
    if (clist_max == clist_ptr) /* Increment the size of the list */
    {
        clist_max = clist_ptr + 1; /* If too big */
        printf("Error in Charlist_place_new_char(): Character list is FULL!\n");
        exit(1);
    }
    /* Place new in the list character */
    Charlist_locate_character(clist, newchar);
    /* This routine repositions a character in the charlist. It removes
     * the character, recalculates loc and inserts it in the list accordingly
     */
    void Charlist_relocate_char(clist, ch)
        struct Charlist *clist;
        struct Character *ch;
    {
        /* Recalculate the position of the character */
        Charlist_recalculate_loc(ch);
        return;
    }
    if (clist_max < 2) /* If it's the only item in the list */
        return;
    /* Remove the character from the list */
    if (ch_ptr == 0) /* If list on list */
    {
        clist_ptr = clist_ptr + 1; /* If list on list */
    }
    else {
        ch_ptr = ch_ptr + 1;
        if (ch_ptr == 0)
            ch_ptr = clist_ptr;
    }
    /* Place the character in the list */
    Charlist_locate_character(clist, ch);
}

/* .....
 * Insert a character after doing a cut */

```

【図144】

```

/* This routine will insert the character beyond the charlist (chrlist).
 * loc is not used for both character and the new character, however
 * place the new character after chrlist. After finished, it is an
 * extremely good idea to resort the heaplist!
 */
void Charlist_insert_after_ptr(charlist, ch, after)
{
    struct Character *newchar; /* Place the new character after this one */
    struct Character *chrlist; /* Pointer to the new character adding */
    newchar = &list->next; /* Set newchar */
    Charlist_recalculate_loc(newchar);
    Charlist_recalculate_loc(chrlist);
    /* Check for full list */
    if (chrlist->next == NULL) /* Increment the size of the list */
    {
        printf("Error in Charlist_remove_char(): Character list is FULL!\n");
        exit(1);
    }
    newchar->prev = chrlist;
    newchar->next = chrlist->next;
    chrlist->next = newchar;
    if (newchar->next != 0)
        newchar->next->prev = newchar;
}

/* This routine will remove the character from the list.
 * chrlist is not used for both character and the new character, however
 * place the new character after chrlist. After finished, it is an
 * extremely good idea to resort the heaplist!
 */
void Charlist_remove_char(charlist, ch)
{
    struct Character *chrlist;
    struct Character *newchar;
    /* Remove the character from the list */
    if (chrlist->next == 0) /* If it is on list */
        chrlist->next = chrlist->first->next;
    else
        chrlist->first->prev = 0;
    newchar->next = chrlist->next;
    if (chrlist->next != 0)
        chrlist->next->prev = newchar;
}

/* This routine will calculate the position of a character
 * void Charlist_recalculate_loc(char) /* Output is the position */
{
    struct Character *chr; /* The character */
    int i, loc;
    /* Calculate the position of the new character */
    loc = 0;
    for (i = 0; i < chrlist->next; i++) /* Position = average position... */
        loc = (chrlist->next + chrlist->first) / 2;
    loc = loc * i;
    chrlist->loc = loc;
}

```

```

/* This routine will place a character (chr) is not in the charlist:
 * void Charlist_insert_after_ptr(charlist, ch)
 * struct Character *chrlist; /* The character */
 * int loc; /* Used for searching for location */
 * struct Character *newchar; /* The character before p */
 *
 * /* Is the list empty? */
 * if (chrlist->next == 0)
 * {
 *     chrlist->next = 0;
 *     chrlist->first = ch;
 *     return;
 * }
 * /* Possible optimization: Don't start from start of list...
 * start from chrlist->first */
 * loc = chrlist->loc; /* Retrieve position */
 *
 * /* Is it at the beginning of the list? */
 * p = chrlist->first; /* Load p with the first item on the list */
 * if (loc < p->loc) /* If it's the first on the list */
 * {
 *     p->prev = ch;
 *     chrlist->first = p;
 *     return;
 * }
 * /* Set the pointers */
 * p->prev = ch;
 * p->next = chrlist->first;
 * chrlist->first = p;
 * /* Set the loc on the list to this */
 * while (p->loc <= loc)
 * {
 *     last_p = p;
 *     p = p->next;
 *     if (p == 0)
 *         break;
 * }
 * /* Remember last position */
 * /* Get the next character */
 * /* If it's the end of list, insert it there */
 * while (p->loc <= loc)
 * {
 *     last_p = p;
 *     p = p->next;
 *     if (p == 0)
 *         break;
 * }
 * /* Insert new position after last_p */
 * last_p->next = ch;
 * if (p == 0)
 *     p->prev = ch;
 * }

```

```

/* Allocate memory for list */
size = size * 2;
malloc(&list, sizeof(struct Character));
/* Allocate memory
 * for the list */
/* Set to zero elements */
list->next = 0;

```

【图 146】

—336—

```

// Filename: climage.cop
// This file contains a vast array image utility functions, for a climage
//
//
//
//
// underline this to make limage.c run invisibly (except for error)
//define PRINT_IMAGES_DEBUG

#include "climage.h"
#include "ia.h"

//... CLASS: climage (an array of chars)
//.....
climage::climage()
{
    sized = 0;
    sized_c = 0;
    allocated_sized = 0;
    allocated_sized_c = 0;
    cpl = 1;
}

climage::climage(int x, int c)
{
    sized = 0;
    sized_c = 0;
    allocated_sized = 0;
    allocated_sized_c = 0;
    cpl = 0;
    allocate(x, c);
}

// the problems won't be circum when
// allocate called allocate
allocate(int x, int c)
{
    sized = 0;
    sized_c = 0;
    allocated_sized = 0;
    allocated_sized_c = 0;
    cpl = 0;
    allocate(x, c);
}

climage::climage(climage img)
{
    allocated_sized, sized_c, // Allocate memory for new climage (loads sizes)
    for (int i = 0; i < sized; i++) // Copy old image to new climage
        for (int j = 0; j < sized_c; j++)
            pixel[i][j] = img.pixel[i][j];
}

// before the climage
climage::~climage()
{
    deallocate();
}

//
//
//
//
// Allocate memory for an climage from memory
void climage::allocate(int x, int c)
{
    deallocate(); // Remove current climage
    if (x < 0 || c < 0)
        cerr << "Illegal climage size: " << x << " by " << c << '\n';
    exit(1);
}

//
//
//
//
// Set climage sizes
sized = x;
sized_c = c;
if (x != 0) && (c != 0)

```

【図148】

```

for (i=R1; i<R2; i++) // Copy clipped image
for (j=C1; j<C2; j++)
    temp_pixel[i-R1][j-C1] = pixel[i][j];
allocate(temp_size, temp_sizeC); // Copy temp image to current image
for (i=0; i<temp_h; i++)
    for (j=0; j<sizeC; j++)
        pixel[i][j] = temp_pixel[i][j];
}

// Read in part of a TIFF file
void class::readTIFF(char *filename)
{
    FILE *tagfile; // Input file
    long int tagfile_position; // Output file position (in bytes)
    long int width; // Width of picture
    long int length; // Length of picture
    long int black; // Value of black pixel
    long int bps; // Bytes per strip
    long int offset; // Start of image data
    long int number; tag_type, int, value; // Tag values
    long int i, j;

    if ((tagfile = fopen(filename, "rb")) == NULL)
    {
        cout << "Unable to open file " << filename << "\n";
        exit(1);
    }
    tagfile_position = 0;

    // Read in header and make sure it's Intel
    i = getBytes(tagfile, tagfile_position);
    if (i != 0x0004)
    {
        cout << "... Error: TIFF File not in Intel Format!\n";
        exit(1);
    }

    // Read in Magic number
    i = getBytes(tagfile, tagfile_position);
    if (i != 0x0002A)
    {
        cout << "... BAD Magic Number for TIFF file!\n";
        exit(1);
    }

    // Find first file directory
    i = getBytes(tagfile, tagfile_position);
    while(tagfile_position < i)
    {
        i = getBytes(tagfile, tagfile_position);
    }

    // Type in directory
    number = getBytes(tagfile, tagfile_position);
    while (number > 0) {
        tag = getBytes(tagfile, tagfile_position);
        type = getBytes(tagfile, tagfile_position);
        length = getBytes(tagfile, tagfile_position);
        value = getBytes(tagfile, tagfile_position);
        if ((len = 1) && (tag & 32768) == 0)
        {
            cout << "Error: Tag length is 1!\n";
        }
    }
}

```

```

// Make the new image (note also adjusted)
for (int i=0; i<temp_h; i++)
    pixel[i][3] = value;
}

// Copy a section of a image into worker, image = reference image
// Range is [R1..R2] [C1..C2]
void class::clipImage(image, int R1, int R2, int C1, int C2)
{
    int r, c, t;

    if (R2 < R1) // Swap R1 & R2
    {
        t = R1; R1 = R2; R2 = t;
    }
    if (C2 < C1) // Swap C1 & C2
    {
        t = C1; C1 = C2; C2 = t;
    }
    // Check limits
    if (R2 < 0) R2 = 0;
    if (R1 < 0) R1 = 0;
    if (C2 < 0) C2 = 0;
    if (C1 < 0) C1 = 0;
    if (R1 > image.height) R1 = image.height;
    if (R2 > image.height) R2 = image.height;
    if (C1 > image.width) C1 = image.width;
    if (C2 > image.width) C2 = image.width;

    allocate(R2-R1, C2-C1);

    for (i=0; i<temp_h; i++)
    {
        for (j=0; j<sizeC; j++)
            pixel[i][j] = image.pixel[i-R1][j-C1];
    }
}

// ClipImage: reduce the size of an image by removing the black
// space around it. This adjusts the image size.
void class::clipImage()
{
    int R1, R2, C1, C2; // The starting and stopping rows of new image
    int i, j;
    struct image temp; // Temporary image
    // Initialise to extremes
    R1 = 0;
    R2 = 0;
    C1 = 0;
    C2 = 0;

    for (i=0; i<temp.h; i++)
    {
        for (j=0; j<temp.w; j++)
        {
            if (pixel[i][j] != 0)
            {
                if (i < R1) R1 = i; // Left edge?
                if (i > R2) R2 = i; // Right edge?
                if (j < C1) C1 = j; // Top edge?
                if (j > C2) C2 = j; // Left edge?
            }
        }
    }

    if (R1 > R2) R1 = R2 = 0; // Check for no pixel found
    if (C1 > C2) C1 = C2 = 0;

    temp.allocate(R2-R1+1, C2-C1+1); // New image is this size
}

```



【図149】

```

    } // while
    while(tagfile_position < raster_offset)
    {
        i = getbytes(tagfile, tagfile_position);
    }
    //----- READ IN FILE -----
    allocate(int* length, (int) width);
    int v;
    for (i = 0; i < length; i++)
    {
        for (j = 0; j < width; j++)
        {
            if ((j&3) == 0)
            {
                v = getc(tagfile);
                tagfile_position++;
                if (v == EOF)
                {
                    cout << "\n Error: Premature End of File found in tiff file\n";
                    exit(1);
                }
                if ((v & 128) == black*255)
                {
                    pixel(i)(j) = 0;
                    if (v == EOF)
                    {
                        pixel(i)(j) = 1;
                        v = v < 1;
                    }
                }
            }
        }
        close(tagfile);
    }
    //-----
    // Read in a byte. File position is updated
    // Error: Premature End of File found in tiff file
    // Error: Premature End of File found in tiff file
    int i;
    int ci;
    if ((ci == EOF) && (error == 1))
    {
        cout << "\n Error: Premature End of File (in getbytes)\n";
        exit(1);
    }
    file_position++;
    return ci;
}
//-----
// Read in an integer from a file. [59, 159]
long int ciange: getbytes(FILE *f, long int* fp)
{
    int ci, c2;
    ci = getbyte(f, fp, 1);
    c2 = getbyte(f, fp, 1);
    return ci + c2*256;
}
//-----
// Read in a long integer
// long int ciange: getbytes(FILE *f, long int* fp)

```

```

    }
    switch (tag)
    {
        case 0x007E: // NewSubFileType
            if (value != 0)
            {
                cout << "Can't Deal With Non-Standard NewSubFileType\n";
                exit(1);
            }
            break;
        case 0x007F: // NewSubFileType
            if (value != 1)
            {
                cout << "Subfile Type not full resolution\n";
                exit(1);
            }
            break;
        case 0x0081: // Width
            width = value;
            break;
        case 0x0082: // Length
            length = value;
            break;
        case 0x0083: // Bits per sample
            if (value != 1)
            {
                cout << "Error: Bits/sample must be 1\n";
                exit(1);
            }
            break;
        case 0x0084: // Compression
            if (value != 1)
            {
                cout << "Compression Used Error\n";
                exit(1);
            }
            break;
        case 0x0085: // Black is
            black = value;
            break;
        case 0x0087: // Thresholding tag
            if ((value < 1) || (value > 3))
            {
                cout << "Thresholding Tag = illegal value\n";
                exit(1);
            }
            break;
        case 0x0091: // Offset to Raster Data
            raster_offset = value;
            break;
        case 0x0092: // hps = value;
            hps = value;
            break;
        case 0x0093: // hps = value;
            hps = value;
            break;
        case 0x0094: // hps = value;
            hps = value;
            break;
        case 0x0095: // hps = value;
            hps = value;
            break;
        case 0x0096: // hps = value;
            hps = value;
            break;
        case 0x0097: // hps = value;
            hps = value;
            break;
        case 0x0098: // hps = value;
            hps = value;
            break;
        case 0x0099: // hps = value;
            hps = value;
            break;
        case 0x009A: // hps = value;
            hps = value;
            break;
        case 0x009B: // hps = value;
            hps = value;
            break;
        case 0x009C: // hps = value;
            hps = value;
            break;
        case 0x009D: // hps = value;
            hps = value;
            break;
        case 0x009E: // hps = value;
            hps = value;
            break;
        case 0x009F: // hps = value;
            hps = value;
            break;
        case 0x00A0: // hps = value;
            hps = value;
            break;
        case 0x00A1: // hps = value;
            hps = value;
            break;
        case 0x00A2: // hps = value;
            hps = value;
            break;
        case 0x00A3: // hps = value;
            hps = value;
            break;
        case 0x00A4: // hps = value;
            hps = value;
            break;
        case 0x00A5: // hps = value;
            hps = value;
            break;
        case 0x00A6: // hps = value;
            hps = value;
            break;
        case 0x00A7: // hps = value;
            hps = value;
            break;
        case 0x00A8: // hps = value;
            hps = value;
            break;
        case 0x00A9: // hps = value;
            hps = value;
            break;
        case 0x00AA: // hps = value;
            hps = value;
            break;
        case 0x00AB: // hps = value;
            hps = value;
            break;
        case 0x00AC: // hps = value;
            hps = value;
            break;
        case 0x00AD: // hps = value;
            hps = value;
            break;
        case 0x00AE: // hps = value;
            hps = value;
            break;
        case 0x00AF: // hps = value;
            hps = value;
            break;
        case 0x00B0: // hps = value;
            hps = value;
            break;
        case 0x00B1: // hps = value;
            hps = value;
            break;
        case 0x00B2: // hps = value;
            hps = value;
            break;
        case 0x00B3: // hps = value;
            hps = value;
            break;
        case 0x00B4: // hps = value;
            hps = value;
            break;
        case 0x00B5: // hps = value;
            hps = value;
            break;
        case 0x00B6: // hps = value;
            hps = value;
            break;
        case 0x00B7: // hps = value;
            hps = value;
            break;
        case 0x00B8: // hps = value;
            hps = value;
            break;
        case 0x00B9: // hps = value;
            hps = value;
            break;
        case 0x00BA: // hps = value;
            hps = value;
            break;
        case 0x00BB: // hps = value;
            hps = value;
            break;
        case 0x00BC: // hps = value;
            hps = value;
            break;
        case 0x00BD: // hps = value;
            hps = value;
            break;
        case 0x00BE: // hps = value;
            hps = value;
            break;
        case 0x00BF: // hps = value;
            hps = value;
            break;
        case 0x00C0: // hps = value;
            hps = value;
            break;
        case 0x00C1: // hps = value;
            hps = value;
            break;
        case 0x00C2: // hps = value;
            hps = value;
            break;
        case 0x00C3: // hps = value;
            hps = value;
            break;
        case 0x00C4: // hps = value;
            hps = value;
            break;
        case 0x00C5: // hps = value;
            hps = value;
            break;
        case 0x00C6: // hps = value;
            hps = value;
            break;
        case 0x00C7: // hps = value;
            hps = value;
            break;
        case 0x00C8: // hps = value;
            hps = value;
            break;
        case 0x00C9: // hps = value;
            hps = value;
            break;
        case 0x00CA: // hps = value;
            hps = value;
            break;
        case 0x00CB: // hps = value;
            hps = value;
            break;
        case 0x00CC: // hps = value;
            hps = value;
            break;
        case 0x00CD: // hps = value;
            hps = value;
            break;
        case 0x00CE: // hps = value;
            hps = value;
            break;
        case 0x00CF: // hps = value;
            hps = value;
            break;
        case 0x00D0: // hps = value;
            hps = value;
            break;
        case 0x00D1: // hps = value;
            hps = value;
            break;
        case 0x00D2: // hps = value;
            hps = value;
            break;
        case 0x00D3: // hps = value;
            hps = value;
            break;
        case 0x00D4: // hps = value;
            hps = value;
            break;
        case 0x00D5: // hps = value;
            hps = value;
            break;
        case 0x00D6: // hps = value;
            hps = value;
            break;
        case 0x00D7: // hps = value;
            hps = value;
            break;
        case 0x00D8: // hps = value;
            hps = value;
            break;
        case 0x00D9: // hps = value;
            hps = value;
            break;
        case 0x00DA: // hps = value;
            hps = value;
            break;
        case 0x00DB: // hps = value;
            hps = value;
            break;
        case 0x00DC: // hps = value;
            hps = value;
            break;
        case 0x00DD: // hps = value;
            hps = value;
            break;
        case 0x00DE: // hps = value;
            hps = value;
            break;
        case 0x00DF: // hps = value;
            hps = value;
            break;
        case 0x00E0: // hps = value;
            hps = value;
            break;
        case 0x00E1: // hps = value;
            hps = value;
            break;
        case 0x00E2: // hps = value;
            hps = value;
            break;
        case 0x00E3: // hps = value;
            hps = value;
            break;
        case 0x00E4: // hps = value;
            hps = value;
            break;
        case 0x00E5: // hps = value;
            hps = value;
            break;
        case 0x00E6: // hps = value;
            hps = value;
            break;
        case 0x00E7: // hps = value;
            hps = value;
            break;
        case 0x00E8: // hps = value;
            hps = value;
            break;
        case 0x00E9: // hps = value;
            hps = value;
            break;
        case 0x00EA: // hps = value;
            hps = value;
            break;
        case 0x00EB: // hps = value;
            hps = value;
            break;
        case 0x00EC: // hps = value;
            hps = value;
            break;
        case 0x00ED: // hps = value;
            hps = value;
            break;
        case 0x00EE: // hps = value;
            hps = value;
            break;
        case 0x00EF: // hps = value;
            hps = value;
            break;
        case 0x00F0: // hps = value;
            hps = value;
            break;
        case 0x00F1: // hps = value;
            hps = value;
            break;
        case 0x00F2: // hps = value;
            hps = value;
            break;
        case 0x00F3: // hps = value;
            hps = value;
            break;
        case 0x00F4: // hps = value;
            hps = value;
            break;
        case 0x00F5: // hps = value;
            hps = value;
            break;
        case 0x00F6: // hps = value;
            hps = value;
            break;
        case 0x00F7: // hps = value;
            hps = value;
            break;
        case 0x00F8: // hps = value;
            hps = value;
            break;
        case 0x00F9: // hps = value;
            hps = value;
            break;
        case 0x00FA: // hps = value;
            hps = value;
            break;
        case 0x00FB: // hps = value;
            hps = value;
            break;
        case 0x00FC: // hps = value;
            hps = value;
            break;
        case 0x00FD: // hps = value;
            hps = value;
            break;
        case 0x00FE: // hps = value;
            hps = value;
            break;
        case 0x00FF: // hps = value;
            hps = value;
            break;
    }
}

```

【図150】

```

long int c1, c2;
c1 = get2bytes(f, fp);
c2 = get2bytes(f, fp);
return c1 + c2*65536;
}

////////////////////////////////////
// Plot image using Flash Graphics at (xloc, yloc) with color
void cimage::plot(int x, int y)
{
    int px, py;           // Pixel coords on the screen (upper left)
    char color;
    for (py=0; py<sizeR; py++)
        for (px=0; px<sizeC; px++)
            color = pixel[py][px];
            if (color & 1)
                xa_plotxy(x+px, y+py); // Draw it
}

////////////////////////////////////
// Plot image with a box around it (with color 1)
void cimage::plotbox(int x, int y)
{
    int px, py;
    for (px=0; px<sizeC+2; px++) { // Draw horizontal lines
        if ((x+px)%42 == 0)
            xa_plotxy(x+px, y);
        if ((x+px+y+sizeR+1)%42 == 0)
            xa_plotxy(x+px, y+sizeR+1);
    }
    for (py=1; py<sizeR+1; py++) {
        if ((x+py)%42 == 0)
            xa_plotxy(x, y+py);
        if ((x+sizeC+1+y+py)%42 == 0)
            xa_plotxy(x+sizeC+1, y+py);
    }
    plot(x+1, y+1); // Plot the image
}

```

【図167】

```

extern void cut_dash(cimage& line, struct Charlist* clist);
cut_dash(("line_to_process", clist);
#endif

#ifdef COMBINING
charlist_combine(("line_to_process", clist);
#endif

#ifdef PRINT_FINAL_CHARLIST
// Print the characterlist
xa_clr_win();
print_charlist(("line_to_process", clist, 10, 10);
xa_flush();
#endif

#ifdef DISPLAY
xa_flush();
#endif

#ifdef DISPLAYING_FINAL_SEGMENTATION
printf("Displaying Final Segmentation. Enter 0 to continue? ");
int dummy;
dummy = auto_input_int();

zero_vpn_terminate(&masteraslist); // *** DELETE ***
}

```

【図151】

```

//
// Filename: cimage.h
// This is the header file for cimage.C
//
////////////////////////////////////
//
// Define KDD_DEBUG 1
#include<stream.h>
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

class cimage; // Forward reference

//-----
// CLASS: cimage
//-----
class cimage
{
public:
    int sizeR; // Number of rows (first index of array)
    int sizeC; // Number of cols. (second index of array)
    int allocated_sizeR; // Amount malloced
    int allocated_sizeC; // Amount malloced
    int dpi; // Dots per inch of the image (0 if unknown)
    unsigned char *pixel; // The binary image (2d array)

    cimage(); // Initialise
    ~cimage(); // Destroy
    cimage(int r, int c); // Initialize with a size
    cimage(cimage& temp);

    void reallocate(int R, int C); // Will allocate memory for an image
    void fill(char value, int R, int C); // Fill image with a constant value
    void clip(cimage& image, int R1, int R2, int C1, int C2);
    void clip_space(); // Clips away space surrounding image
    void readtiff(char *filename); // Read in a tiff image
    void plot(int x, int y); // Plot binary image
    void plotbox(int x, int y); // Plot with a box around it

private:
    void allocate(int r, int c); // Allocate memory
    void deallocate(); // Deallocate memory
    int get1byte(FILE *f, long int& file_position, int noerror);
    long int get2bytes(FILE *f, long int& fp);
    long int get4bytes(FILE *f, long int& fp);
};

```

【図152】

```

// Filename: combine.c
// Routine for combining incorrectly separated characters
//
// =====
#include <charlist.h>
#include <math.h>

#define VERY_SMALL -32000 // Just for debugging!! DELETE EVENTUALLY
#define FRAC_VALID_DIST 9.0 // How far apart breaks must be (in % pixels)
// =====
// Compute the overlapping ratio. Multiply this
// By the smaller dot to get ranges
// How much longer the body of the l must be
// than the dot (body > x * dot)
#define DOT_TO_BODY_RATIO 2 // The ratio of the dot the line should
// have to the body of the line
#define PERCENT_RATIO 0.16 // The ratio of dot width / line width
// =====
#define PORT_TO_DOT_RATIO 0.7 // Minimum ratio of dot width / line width
// =====
#define PORT_TO_DOT_RATIO 0.1 // Ignore top and bottom list for computing the
// slope of the line in a percent slope.
#define PORT_PERCENT_OVERLAP 0.18 // Multiply this by width of line of s to see
// how far the characters must overlap
// =====
#define PORT_DOT_WIDTH_MIN 5.50 // Minimum ratio of dot width / line width
// =====
#define PORT_DOT_WIDTH_MAX 1.10 // Minimum ratio of dot width / line width
// =====
#define PORT_DOT_HEIGHT_MIN 0.43 // Minimum ratio of dot height / line height
// =====
#define PORT_DOT_HEIGHT_MAX 0.70 // Minimum ratio of dot height / line height
// =====
#define PORT_DOT_SLOPE 0.37 // Minimum inverse slope of the line (aka
// inverse of maximum slope of line)

// Define a Combinelist which is a list of 4 characters (with information
// stored about them) that are looked at to see if we need to combine them.
struct Combinelist {
    struct Character *c1; // The first character in the list
    struct Character *c2; // The second character in the list
    // .. (and so on) 10 = none)
    struct Character *c4; // The next character to load in
    int b1; // 1's bottom distance between 1 & 2
    int b2; // 2's bottom distance between 2 & 3
    int d1; // 1's distance from top of 3 to bottom of other
    int d2; // 2's distance from top of 3 to bottom of other
    int d3; // (defined from top of 3 to bottom of other)
    int d4; // (VERY_SMALL = undefined)
};

// Define external variables and routines (for displaying and debugging)
extern int cursor_x;
extern int cursor_y;
extern print_character(char, struct Character *chr, int x, int y);
extern print_charlist(struct Combinelist, struct Character *clist, int x, int y);

// =====
// Character debugging routines
// =====
void debug_print_line(
    struct Character *clist, // base of the line
    struct Combinelist *comb // The list
) {
    int x, y;
    x = y = 10;
}

```

```

x = list_vln();
if (comb->c1 != 0)
    print_character(line, comb->c1, x, y);
x += 10 - comb->c1->left + comb->c1->right;
if (comb->c2 != 0)
    print_character(line, comb->c2, x, y);
x += 10 - comb->c2->left + comb->c2->right;
if (comb->c3 != 0)
    print_character(line, comb->c3, x, y);
x += 10 - comb->c3->left + comb->c3->right;
if (comb->c4 != 0)
    print_character(line, comb->c4, x, y);
x += 10 - comb->c4->left + comb->c4->right;
print_charlist(line, clist, 10, 20, clist->vector_height);
xs_flush();

// =====
// Character combining routines
// =====
// This routine returns if two characters are above each other.
// Returns 1 for overlapping or character does not exist, 1 for 1st higher,
// 0 for 2nd higher, -1 if also updates list.
int comb_height_data(
    struct Character *c1, // The first character
    struct Character *c2, // The second character
    int *h1, // The height (updated)
    int *h2 // If it's been calculated,
    // don't recalculate it
    // If character doesn't exist
    // Return no gap exists
) {
    if (c1 == 0)
        return 0;
    if (c2 == 0)
        return 0;
    if (c1->bot <= c2->top)
        // If bottom of c1 is higher than top of c2
        return 1;
    if (c2->bot <= c1->top)
        // If bottom of c2 is higher than top of c1
        return 2;
    if (c1->bot < c2->top)
        // Return the gap
        return 1;
    if (c2->bot < c1->top)
        // Return the gap
        return 2;
    if (c1->bot < c2->top)
        // Return no gap exists
        return 0;
    if (c2->bot < c1->top)
        // Return no gap exists
        return 0;
}

```

【図153】

```

for (l1=c1->bot - 4; l1 < c2->bot; l1++)
    botpos += c2->curve2(l1) + c2->curve2(l1);
}

// Compute position and exit
*d12 = botpos - toppos;

if (c1->float < *d12) / ((float) width_small);
return 1;
else
    return 0;
}

// This routine is called from comb_dist() with a pair of characters
// that probably isn't as close as it seems.
// Characters. This determines if close enough to be combined.
int comb_dist_notif()
{
    struct character *c1; // The first character
    struct character *c2; // The second character
    int h1; // The height of the first character
    int h2; // The height of the second character
    int width_small; // The width of the smaller character

    int toppos, botpos; // Positions of the top and bottoms
    float f;

    if (h12 == 1)
    {
        // Case when c1 is higher than c2
        // Find bottom pos of c1
        if ((c1->bot - c1->top) < 4) // If character very small, use .loc
            botpos = c1->loc;
        else
        {
            toppos = 0;
            for (l1=c1->bot - 4; l1 < c1->bot; l1++)
                botpos += c1->curve2(l1) + c1->curve2(l1);
        }

        // Find top pos of c2
        if ((c2->bot - c2->top) < 4) // If character very small, use .loc
            toppos = c2->loc;
        else
        {
            toppos = 0;
            for (l1=c2->top; l1 < c2->top + 4; l1++)
                toppos += c2->curve2(l1) + c2->curve2(l1);
        }

        // Compute position and exit
        *d12 = toppos - botpos;

        f = ((float) *d12) / ((float) width_small);
        if (f < FRAC_VALID_DIST)
            return 1;
        else
            return 0;
    }

    // Case when c2 is higher than c1
    // Find bottom pos of c1
    if ((c1->bot - c1->top) < 4) // If character very small, use .loc
        botpos = c1->loc;
    else
    {
        toppos = 0;
        for (l1=c1->bot - 4; l1 < c1->bot; l1++)
            botpos += c1->curve2(l1) + c1->curve2(l1);
    }

    // Find top pos of c2
    if ((c2->bot - c2->top) < 4) // If character very small, use .loc
        toppos = c2->loc;
    else
    {
        toppos = 0;
        for (l1=c2->top; l1 < c2->top + 4; l1++)
            toppos += c2->curve2(l1) + c2->curve2(l1);
    }

    // Compute position and exit
    *d12 = botpos - toppos;

    f = ((float) *d12) / ((float) width_small);
    if (f < FRAC_VALID_DIST)
        return 1;
    else
        return 0;
}

// Find bottom pos of c2
if ((c2->bot - c2->top) < 4)
    botpos = c2->loc;
else
    botpos = 0;
}

```

【図154】

```

else
{
    toppos = 0;
    for (i=c1-stop; i < c1-stop + 4; i++)
        toppos += c1-curve2[i] + c1-curve2[i+1];
}
// Find bottom pos of c2
if ((c2-bot - c2-stop) < 4)
    botpos = c2-stop; // If character very small, use .loc
else
{
    botpos = 0;
    for (i=c2-bot - 4; i < c2-bot; i++)
        botpos += c2-curve2[i] + c2-curve2[i+1];
}
// Compute position and exit
*d1 = botpos - toppos;
if (i==(float) *d12) / ((float) w(c2)_small);
if (i < FRAC_VALID_DIST)
    return 1;
else
    return 0;
}

// This routine returns two character widths and heights
// to the caller. It computes characters from top of lower one
// to bottom of higher one. Returns 0 - Bad. 1 - Good. Also updates d12
// Position is generated to be relative to bot height
int comb_dist(
    struct Character *c1, // The first character
    struct Character *c2, // The second character
    int *d1, // The first character is higher
    int *d12) // The last value
{
    // The size of c1 (width)
    int width_c1;
    // The size of c2
    int width_c2;
    // The size of the smaller character
    int width_small;
    // The height of c1
    int height_c1;
    // The height of c2
    int height_c2;
    float f;

    width_c1 = c1-right - c1-left;
    width_c2 = c2-right - c2-left;
    if (width_c1 < width_c2)
        width_small = width_c1;
    else
        width_small = width_c2;
    if (*d12 != VAR_MAX) // If it's been computed, don't recompute it
    {
        f = ((float) *d12) / ((float) width_small);
        if (f < FRAC_VALID_DIST)
            return 1;
        else
            return 0;
    }
    height_c1 = c1-bot - c1-stop;
    height_c2 = c2-bot - c2-stop;
    if (h12 == 1) // c1 is higher than c2
}

```

```

// This routine uses if larger characters are overlapping enough (left and
// right wise.) used for identifying type 1 percent signs.
// Returns a 1 if overlapping for a type 1 percent, 0 otherwise
// Assumes that the characters ARE overlapping!
int comb_dist_pos(
    struct Character *c1, // The first character
    struct Character *c2, // The second character
    int i1, i2, // The length of the characters
    int a1, a2, // The left sides of the character
    int b1, b2, // The right sides of the character
    int swap) // MAC var
{
    i1 = c1-right - c1-left;
    i2 = c2-right - c2-left;
    if (i1 > i2)
    {
        swap = c1;
        c1 = c2;
        c2 = c1;
        swap = i1;
        i1 = i2;
        i2 = swap;
    }
    if ((float) i1 / (float) i2 < PERCENT_TYPE_MAX)
        return 0;
    if ((float) i1 / (float) i2 > PERCENT_TYPE_MAX)
        return 0;
    if (c2-left < c1-left)
        a1 = c2-left;
        a2 = c1-left;
    if (c2-right > c1-right)
        a2 = c1-right;
        a1 = c2-right;
    if ((float) (a2 - a1) / ((float) i1) > PERCENT_TYPE_MAX)
        return 1;
    else
        return 0;
}

```

【図155】

```

// This routine will check for the two characters below the list
// part of a percent sign o/ Returns 1 for it's a %, 0 for no.
int comb_percent(
    struct Character *c1,
    struct Character *c2)
// The characters

// Check for o/ part of percent (types 2)
float hl, ul;
float hl, ul, wd;
int p;

hl = (float) (c1->bot - c1->stop);
ul = (float) (c1->right - c1->left);
wd = (float) (c1->right - c1->left);

if (ul * PORT_PERCENT_DOTLLX <= (float) (c1->right - c1->left))
if (ul * PORT_DOT_WIDTH_ULX <= wd) // If dot is width is not too small
if (ul * PORT_DOT_HEIGHT_ULX <= hl) // If dot is width is not too large
if (hl * PORT_DOT_HEIGHT_ULX <= wd) // If dot is width is not too small
if (hl * PORT_DOT_HEIGHT_ULX <= wd) // If dot is width is not too large

{
    p = (int) (wd * PORT_DOT_HEIGHT_ULX); // Ignore from top and bottom
    { (float) (c1->bot - c1->stop) - c1->curved(c2->bot - p) /
        (float) (c1->bot - c1->stop - p - p) * PORT_MIN_SLOPE)
        return 1;
    }
    return 0;
}

// This routine will check for the two characters being the list
// part of a percent sign o/ Returns 1 for it's a %, 0 for no.
int comb_right_percent(
    struct Character *c1,
    struct Character *c2)
// The characters

// Check for o/ part of percent (types 2)
float hl, ul;
float hl, ul, wd;
int p;

hl = (float) (c1->bot - c1->stop);
ul = (float) (c1->right - c1->left);
wd = (float) (c1->right - c1->left);

if (ul * PORT_PERCENT_DOTLLX <= (float) (c1->right - c1->left))
if (ul * PORT_DOT_WIDTH_ULX <= wd) // If dot is width is not too small
if (ul * PORT_DOT_HEIGHT_ULX <= hl) // If dot is width is not too large
if (hl * PORT_DOT_HEIGHT_ULX <= wd) // If dot is width is not too small
if (hl * PORT_DOT_HEIGHT_ULX <= wd) // If dot is width is not too large

{
    p = (int) (wd * PORT_DOT_HEIGHT_ULX); // Ignore from top and bottom
    { (float) (c1->bot - c1->stop) - c1->curved(c2->bot - p) /
        (float) (c1->bot - c1->stop - p - p) * PORT_MIN_SLOPE)
        return 1;
    }
    return 0;
}

// This routine will slide the list over, resulting in a new character
// i.e. 1 <= 2 <= 3 <= 4 <= New (entire list)
void comb_slide(
    struct Combinelist *comb) // The list

// Slide the characters down
comb->c1 = comb->c2;
comb->c2 = comb->c3;
comb->c3 = comb->c4;
comb->c4 = comb->c5;
if (comb->nchar == 0)
    comb->nchar = comb->nchar + 1; // Get the next character in the list
comb->h12 = comb->h13; // Set down heights as well
comb->h13 = comb->h14; // Set to undefined
comb->h14 = -1; // Set down distances
comb->c1 = comb->c2;
comb->c2 = comb->c3;
comb->c3 = comb->c4;
comb->c4 = comb->c5;
comb->h12 = comb->h13;
comb->h13 = comb->h14;
comb->h14 = -1; // Set to undefined
comb->h12 = comb->h13;
comb->h13 = comb->h14;
comb->h14 = -1; // Set to undefined

// This routine will combine two chars (which better be adjacent!) This
// works by removing c3 and adding it on to the end of c1.
void comb_combine(
    struct Character *c1,
    struct Character *c2)
// The first character
// The second character

int loc; // The location of the new character
int top; // The top of the new character
int bot; // The bottom of the new character
int bot1; // The bottom of the first character
int bot2; // The bottom of the second character
int al1; // Range of overlapping portion of characters
int al2; // The left and right of new character
int l; // The left and right of new character

// Combine c1 and c2, and compute top and bot
if ((c1->bot - c2->stop) <= (c1->bot - c1->stop))
{
    // Nonoverlapping chars
    for (l=c2->stop; l<=c2->bot; l++) // Copy curves of c2 into c1
    {
        c1->curved[l] = c1->curved[l];
        c1->curved[l+1] = c1->curved[l];
    }
    // compute new location
    loc = (c1->loc + (c1->bot - c1->stop) + c2->bot - c2->stop) /
        (c1->bot - c1->stop + c2->bot - c2->stop);
    ploc = loc * 100 / 100;
    // Fill blank space between to chars (set to loc, and loc won't change.)
    // (and set top and bot)
    if ((c1->bot - c2->stop)
    {
        for (l=c2->bot; l<=c2->stop; l++)
        {
            c1->curved[l] = c1->curved[l];
            top = c1->stop;
        }
    }
}

```

【図156】

```

1: c1->left < c2->left;
   left = c1->left;
else
   left = c2->left;
   if (c1->right < c2->right)
       right = c1->right;
   else
       right = c2->right;

// Load information about the stuff.
c1->type = c1;
c1->top = top;
c1->bot = bot;
c1->left = left;
c1->right = right;
c1->next = c2->next; // Skip over c2
if (c1->next == 0)
    c1->next = c2;
c1->type = CHARACTER_CONJURED;
c2->type = CHARACTER_ERASED;
}

// This routine will combine characters 1 and 2, and slide them
// out
void comb_combine12()
{
    struct comblist *comb; // The list
    comb_combine(comb->xc1, comb->xc2);
    comb_slide(comb);
}

// The main routine
// This routine will combine characters 1 and 2, and slide them
// out
void charlist_combine()
{
    struct charlist *clist; // Image of the line
    clist->next = 0; // The character list
    struct comblist *comb; // The 4 characters in the que we're evaluating
    int dummy;

    // Initialise the combination list
    comb->next = clist->first; // slide in 4 characters into the list
    comb_slide(comb);
    comb_slide(comb);
    comb_slide(comb);
    comb_slide(comb);

    // Main loop: Warning: indentation is not standard (but not too bad)
    while(comb->c2 != 0) // Loop until list is empty
    {
        // Combine 1 & 2
        if (comb->height_delta(comb->c1, comb->c2, &comb->h12))
            if (comb->dist(comb->c1, comb->c2, comb->h12, &(comb->d12)))
            {
                // Combine 3 & 3? Instead!
                if (comb->c3 != 0)
                {
                    if (comb->height_delta(comb->c1, comb->c3, &(comb->h13)))
                        if (comb->dist(comb->c1, comb->c3, comb->h13, &(comb->d13)))
                            if (comb->d13 < comb->d12)
                            {

```

[illegible]



【☒ 1 5 8】

[illegible]

—348—

[illegible]

【図160】

```

//def PRINT_INT0
int dummy;

histogram = ivector(line.size()); // Allocate the memory
while (charen != 0) // Loop until end of list
{
    cd_char(line, clist, lcharon, histogram);
} //def PRINT_INT0
printf("Enter a to continue\n");
scanf("%d", &dummy);
charen = charon->next; // Get the next character
free_vector(histogram); // Deallocate the memory
}

// Check for the dash between a character, and break it if needed
void cd_char(
// The line
// Point to charlist
// Point to the character we're on (modified
// to skip past newly inserted divisions)
// The histogram (previously allocated)
int *hist)
{
    int p;
    // Offset if plural width is too small.
    if ((ch->right - (ch->left) < MIN_ABS_WIDTH_TO_CONSIDER)
        return;
} //def PRINT_INT0
re_char();
printf("%c", character.info);
printf(" top/bot = (%d, %d)\n", (ch->stop, (ch->bot));
printf(" left/right = (%d, %d)\n", (ch->left, (ch->right));
}

// Compute the histogram
// histogram (line, ch, hist)
// Break left side
// cd_half_dash_out(line, ch, hist);
if (p != 0)
{
    // Out the character
    // Out the histogram
    printf("left dash out\n");
    printf("%c", character.info);
    printf(" top/bot = (%d, %d)\n", (ch->stop, (ch->bot));
    printf(" left/right = (%d, %d)\n", (ch->left, (ch->right));
}

// Break right side
p = cd_half_dash_out(line, ch, hist);
if (p != 0)
{
    // Out the character
    //def PRINT_INT0
    printf("right dash out\n");
}

// MAIN ROUTINE
// This will cut the dashes of characters
// The image
// The current charlist
// The area of the histogram
// The character we're processing

```

```

// filename: level2d.c
// This file contains the routines to do level 3 segmentation (slightly
// tweaking) using method 2 (three-pass approach)
//
// The includes
#include "level2d.h"
#include "globals.h"
#include "images.h"
#include "myalloc.h"

// The constants
#define MIN_PIXELS_THRESHOLD 3
#define HISTOGRAM_THRESHOLD 10

// This routine will adjust left, right, curvel, top, bottom
// and pos to represent a new character in the charlist, then it
// calls classify_image to replace the old character
// Note that the first time through the while loop a character is
// added, so shrinking up one side is 'in vels'
void l3l_shrinkup(char *
    changed_line,
    struct Character *ch)
{
    int top, bot, left, right; // The new left and right stuff.
    int i; // pos = position; loc = pos + 8
    int loc, loc_count, pos; // Shrinklet; set to {'ch'-curvel and 2
    int flag; // flag

    curvel = {'ch'-curvel;
    curvel = {'ch'-curvel;

    // Adjust the top of the character. If it needs to move down
    flag = 0; // top found when flag == 1;
    for (top = ('ah')-stop; top < {'ch'}-stop; if (flag == 0); top++)
        if ((line.pixel[top][c] & 1)) // stop at the first on pixel
            {
                flag = 1;
                break;
            }

    // Adjust the bottom of the character
    flag = 0; // bot found when flag == 1;
    for (bot = {'ch'}-start; bot > {'ch'}-stop) as (flag == 0); bot--)
        if ((line.pixel[bot][c] & 1)) // stop at the first on pixel
            {
                flag = 1;
                break;
            }

    // Now adjust curvel and curvel, updating left and right to 'shrink'
    // tightly around the border of the character (move the borders up to the
    // edge of the characters.
}

```

```

loc = loc_count + 0; // for position of the char (on the fly)
for (r=top; r<bot; r++)
while (curvel[r] < curvel[r+1]) // Loop until they're equal
if ((line.pixel[r][curvel[r]] & 1)) // If there is a non-empty space
break; // Then exit (left border not moved)
else
curvel[r] ++; // There's an empty space to the right
while (curvel[r] < curvel[r+1]) // Slide the border over to meet it
if ((line.pixel[r][curvel[r]-1] & 1)) // Do the same with the right border:
break;
else
curvel[r] --;

if (curvel[r] != curvel[r+1]) // there is a pixel on this line...
{
    loc = curvel[r] + curvel[r];
    loc_count++;
}
if (loc_count != 0)
{
    loc = loc + 4 / loc_count; // computes the position (used for .loc)
    pos = loc/r; // Compute the real position

    // Find left and right
    left = curvel[top];
    right = curvel[top];
    for (r=top; r<bot; r++)
    {
        if (curvel[r] < left)
            left = curvel[r];
        if (curvel[r] > right)
            right = curvel[r];
    }

    if (loc == 0)
    {
        loc = 4 * (left+right)/2;
        pos = (left+right)/2;
    }

    for(r=top; r<bot; r++) // Fill in the blank lines with pos
        if ((line.pixel[r][pos] & 1))
            curvel[r] = curvel[r] + pos;

    // copy all the new attributes in
    {'ch'}-loc = loc;
    {'ch'}-stop = top;
    {'ch'}-start = bot;
    {'ch'}-left = left;
    {'ch'}-right = right;
}

// This routine will cut "ch" into two at point p.elist is updated.
// and "ch" returns so the "ch" points to the right half of the character
void cut_ch(struct Character **
    elist, int p)
{
    // The character list
    struct Character *elist;
    // The character to be cut
    struct Character *ch, // where to cut the character
    int p;
}

```

【図162】

```

// Where the minimum value is
// Left and right boundaries of where to search
int min, min_at;
int i;

// Compute the histogram
132_histogram[line, ch, hist];

// start at left side
p = ('ch' - 'left');
while (p < p_max)
{
    while (p < p_max)
    {
        if (hist[p] > histogram_threshold)
        {
            break;
        }
        // This is the left side of the area to search
        1 = p;
        while (p < p_max)
        {
            if (hist[p] > histogram_threshold)
            {
                break;
            }
            // Remember right edge;
            r = p;
            if (p < p_max)
            {
                // If we haven't gone over the edge
                min = hist[1];
                min_at = 1;
                for (i = 1; i < r; i++)
                {
                    if (hist[i] < min)
                    {
                        min = hist[i];
                        min_at = i;
                    }
                }
                // If the min is small enough, cut there
                // (min < min_cut_threshold)
                132_cut_character(line, clist, ch, min_at); // Cut the character
                // Note: ch is changed to the right char,
                // which shouldn't mess things up
                printf("Level 3 made a cut.\n");
            }
        }
    }
}

// This will do histogram touching on the characters for level 2 method 3
// MAIN ROUTINE
// .....
// This will do histogram touching on the characters for level 2 method 3
void level2(cimage_t clist,
            struct Charlist* clist) // The cimage
{
    int histogram; // The area of the histogram
    struct Character* charon; // The character we're processing
    histogram = (vector(line, size)); // Allocate the memory
    charon = clist->first;
    while (charon != 0)
    {
        132_insert_break(line, clist, charon, histogram, MIN_CUT_THRESHOLD);
    }
}

```

```

// The new character
int i;
newchar = &clst->charlist->newchar; // Set newchar to point to the end
// of the charlist (the next blank character)

for (i = 0; i < clist->height; i++) // Loop through the char (up-down)
{
    newchar->curved[i] = ('ch' - 'curve2[i]); // Break the character
    if (newchar->curved[i] < p) // Make sure curves of new char are > p
    {
        newchar->curved[i] = p;
    }
    if (newchar->curved[i] > p) // Make sure curves of old char are <= p
    {
        ('ch' - 'curve2[i]) = p;
    }
}

newchar->top = ('ch' - 'top;
newchar->bot = ('ch' - 'bot;
newchar->left = ('ch' - 'left;
newchar->right = ('ch' - 'right;

132_subintrap_char(line, newchar); // Place newchar into the charlist
132_subintrap_char(clist);
Charlist_replace_char(clist, 'ch');

*ch = newchar; // Adjust ch to the new character
}

// .....
// Compute the histogram
// The line
// Pointer to the character we're on (modified)
// To skip past newly inserted divisions)
// The histogram array (output)
int *hist;

int i;
// Save the histogram
for (i = 'ch' - 'left; i < ('ch' - 'right; i++)
{
    hist[i] = 0;
}
for (i = 'ch' - 'top; i < ('ch' - 'bot; i++) // Compute histogram
{
    for (j = 'ch' - 'curve2[i]; j < ('ch' - 'curve2[i] + 1)
    {
        hist[i]++;
    }
}

// Check if a character need to be broken, and break it if so
void 132_insert_break(
    cimage_t clist,
    struct Character* charon,
    struct Character* ch,
    int *hist,
    int min_cut_threshold,
    int histogram_threshold)
{
    int p;
    int p_max;
}

```

—352—

[illegible]

1

—353—

[illegible]



—355—

```

// Find Projection at a rotated angle
// Note: This routine is extremely unoptimized for speed and space!
// Discrete Breaklist, Num is the break on the left side to start
// Output: Image, theta,
// allocated a highest index allocated for a
// Index_0a = VPP, max_a = highest VPP index value (output),
// Index_0a = what point a[0] corresponds to.
// Define P_STORAGE 1000 // Memory allocated for projection
// Define P_STORAGE 1000 // Where the zero is
// void findproj(image, angle, theta, output image)
// Double theta, angle, angle, angle of the rotated delta (pi/2 - p/2)
// struct Breaklist* blist, // The breaklist for the image
// int NumBl, // break # of left side for computing profile
// int NumBl, // break # of right side for computing profile
// int NumBl, // Right point to match for
// int NumBl, // Right point to match for
// int NumBl, // Output: The profile
// int NumBl, // Output: What point a[0] corresponds to
// int Index_0a
// int store[P_STORAGE], // The profile
// int start_store, stop, // The highest values used in store
// int E; // Misc. Vars
// int D;
// if (store[P_STORAGE] == 0) // The start and stop locations
// int rightbreak = new int[image.size()];
// // Initialize for computing the Angular VPP
// for (i=0; i < P_STORAGE; i++) store[i] = 0;
// start_store = P_STORAGE; // Define the bounds to zero
// breaklist_return_curves(blist, NumBl, NumBl, leftbreak, rightbreak);
// // Compute the Angular VPP
// for (i=0; i < image.size(); i++) // Loop through image
// for (leftbreak[i]; leftbreak[i]; i++) // Loop through image
// if (i < (image.size() - i * tan(theta))) // If it's outside the line
// {
// break;
// }
// // End: From looping through c
// {
// image.plot(x[i], c[i] = 0);
// }
// p = VPP.compute_point(c, theta); // Calculate point
// if (p < 0) // (p > P_STORAGE) // Check range
// {
// cout << "Error in find_projection()! P_STORAGE too small\n";
// exit(1);
// }
// store[p];
// if (p < start_store) start_store = p; // Increment count
// if (p > stop_store) stop_store = p;
// }
// }
// // Deallocate memory used
// delete (image.size) leftbreak;
// delete (image.size) rightbreak;
// // Copy store to a with information
// if (start_store < P_STORAGE) // (stop_store > 0) // If no projection
// {
// max_a = 0; // Return size of a

```



[illegible]

【図171】

```

allocated_a, a, max_a, index_of_a;

// Find maximum projection
max_value_a = a[0];
for (i=1; i<max_a; i++)
    if (a[i] > max_value_a)
        max_value_a = a[i];

// Choose range to ignore points before first peaks on left and right
z1 = 0;
z2 = max_a-1;

if (theta != 0)
{
    // Find left peak (while slope is still rising), and above LA_CROSS_HEIGHT
    while (a[z1] < a[z1+1]) || (a[z1] < a[z1+2]) || (a[z1] < a[z1+3]) ||
        (a[z1] < LA_CROSS_HEIGHT*max_value_a)
    {
        if (i-->0) -- max_a - 3; // If got to left edge
        min = min_at = -1; // Return no minimum found
        free_vector(a);
        return;
    }

    // Find right peak (while slope is falling (free right to left))
    while (a[z2] < a[z2-1]) || (a[z2] < a[z2-2]) || (a[z2] < a[z2-3]) ||
        (a[z2] < LA_CROSS_HEIGHT*max_value_a)
    {
        if (i--<0) ++ z1; // If reached the left point
        min = min_at = -1; // Return no minimum found
        free_vector(a);
        return;
    }

    find_minimum_in_vector(a, z1, z2, min, min_at); // Find the minimum

    // ***** The above is mostly copied from void find_min_valley_at_theta *****
    // ***** The following section is VERY VERY poorly written!!!!

    for (i=cursor_x-1; i<cursor_col_right-1; i++) // Draw line above
        cursor_y = 8;

    // Plot the image
    print_image_with_breath(image, blast, bmax1, bmax2, cursor_x, cursor_y);

    // Find position of the C line and plot it
    struct Break bl; // Find initial offset
    Breaklist_return_break(blast, bmax1, bmax2); // Get first break
    int cx = cursor_x - bl.loc/image_size + c + 2;
    int cy = cursor_y + image_size - 2;
    plot_line_at_angle(C, cx, cmax1, cursor_x - 4, cy + 5); // Right line

    cursor_x += image_size - 5+4;

    // Plot the histogram
    int ttemp = 0;
    plot_histogram(blast, max_a, cursor_x, cursor_y);
    print_image_with_breath(image, blast, bmax1, bmax2, cursor_x, cursor_y);
    max_value_a = max_a - 3; // Plot dotted line
    if (theta != 0)
    {
        int tct = (int) (LA_CROSS_HEIGHT * (double) max_value_a);

```

```

        for (i=cursor_x-5; i<cursor_x+max_a+4; i++)
        {
            if (i%8 == 0)
                x1_plot(y1, cursor_x + ttemp - tct);
            cursor_x += ttemp + 2;

            // Plot array information at bottom
            z1 = cursor_x;
            z2 = cursor_x; // NOT!!!

            if (theta != 0) // Plot bar
            {
                for (i=z1; i<=z2; i++)
                {
                    x1_plot(y1, i+5);
                    x1_plot(y2, i+4);
                    x1_plot(y3, i+3);
                    x1_plot(y4, i+2);
                    x1_plot(y5, i+1);
                }
            }

            // Draw array
            if (theta == 0)
            {
                min_at = bl.loc/image_size + 1;
                z1 = min_at + cursor_x;
                x1_plot(y1, z1);
                x1_plot(y2, z1);
                x1_plot(y3, z1);
                x1_plot(y4, z1);
                x1_plot(y5, z1);
                cursor_x += 10;
            }
            free_vector(a);
        }

        // ***** The find peak and valley routines *****
        // ***** Subroutine to find next peak to the right in a projection *****
        // ***** none found *****
        int find_next_peak_right(
            int proj1, // The projection (an array)
            int pos, // The max size of the projection (not inclusive)
            int min_peak_height, // Where to start looking
            int min_peak_height) // Minimum value for the peak

        {
            int peak; // Where the peak is
            peak = -1; // Set to not found
            while ((pos < proj1_size) && (proj[pos] > min_peak_height))
            {
                if (proj[pos-1] > proj[pos]) pos = pos-1;
                return peak;
            }

            // ***** The find next peak to the left in a projection *****
            // ***** none found *****
            int find_next_peak_left(
                int proj1, // The projection
                int pos, // Where to start looking
                int min_peak_height, // Minimum value for the peak
                int peak) // Where the peak is
            {
                // ***** The find next peak to the left in a projection *****
                // ***** none found *****
                return peak;
            }
        }
    }
}

```

【図172】

```

peak = -1;
// Set to not found
while ((i-pos) > 0) && (peak < -1)
{
    if ((proj[pos+1] <= proj[pos]) && (proj[pos+1] <= min_peak_height))
        min_peak_pos = i;
    return peak;
}

// Subroutine to find next valley to the right in a projection
// -1 = none found
int find_valley_to_right()
{
    int proj_pos;
    // The max size of the projection (not inclusive)
    int max_valley_depth;
    // Highest point a valley can be
    int valley_pos;
    // Where the valley is

    valley = -1;
    // Set to not found
    while ((i-pos) < proj_pos) && (valley < -1)
    {
        if ((proj[pos+1] <= proj[pos]) && (proj[pos+1] <= max_valley_depth))
            valley_pos = pos + 1;
        return valley;
    }

    // Subroutine to find next valley to the left in a projection
    // -1 = none found
    int find_valley_to_left()
    {
        int proj_pos;
        // The projection
        int max_valley_depth;
        // Where to start looking
        int valley_pos;
        // Maximal value for the valley

        int valley;
        // Where the peak is

        valley = -1;
        // Set to not found
        while ((i-pos) <= 0) && (valley < -1)
        {
            if ((proj[pos+1] <= proj[pos]) && (proj[pos+1] <= max_valley_depth))
                valley_pos = pos + 1;
            return valley;
        }

    }

    // Compute the Hebrard's angles given a valley position
    // -1 = none found
    void find_valley_angles()
    {
        int proj_pos;
        // The projection vector
        int valley_pos;
        // The position to the right of the 1st peak
        double thetal;
        // Angle of left wall
        double thetar;
        // Angle of right wall
        int min_peak_height;
        int max_valley_depth;

        int ptx, ply;
        // The coord of left peak
        int ptx, ply;
        // The coord of left valley bottom
        int ptx, ply;
        // The coord of right valley
        int ptx, ply;
        // The coord of right peak

        // Find p1: the peak to the left of the valley
        ptx = find_peak_to_left(proj_valley_pos, min_peak_height);
        if (ptx < -1)
            ptx = 0;
        ply = proj[ptx];
        // Get its height

```

```

// Find p2: the valley to the right of the given peak
ptx = find_valley_to_right(proj_valley_pos, max_valley_depth);
if (ptx < -1)
    ptx = proj_pos + 1;
// If not found, take it the right side
ply = proj[ptx];

// Find p4: the next peak to the right of valley p3
ptx = find_peak_to_right(proj_valley_pos, min_peak_height);
if (ptx < -1)
    ptx = proj_pos + 1;
// If not found, take it the right side
ply = proj[ptx];

// Find p3: the nearest valley to the left of peak at p4
ptx = find_valley_to_left(proj_valley_pos, max_valley_depth);
if (ptx < -1)
    ptx = 0;
// If not found, make it the left side
ply = proj[ptx];

thetal = atan2(double)(ptx-ptx1)/(double)(ply-ply1); // This is positive
thetar = atan2(double)(ptx-ptx1)/(double)(ply-ply1); // This is negative

// Compute the Hebrard's angles given a valley position
// -1 = none found
void compute_angles_threshold()
{
    int proj_pos;
    // The VTP (an array)
    int level;
    // The size of proj
    int min_valley_depth;
    // The threshold level
    int min_peak_height;
    // The outputs

    int max_valley_pos;
    // The maximum value of the projection
    int i;
    double a, b;
    FILE *fp;

    if ((fp = fopen("constants.dat", "r")) == NULL)
    {
        error = "Error in opening constants.dat\n";
        exit(1);
    }

    fscanf(fp, "%lf", &a);
    for (i=0; i<level; i++)
        fscanf(fp, "%lf", &a, &b);
    if ((a <= -1) || (b <= -1))
    {
        max_valley_depth = -1;
        min_peak_height = -1;
        return;
    }
    fclose(fp);

    // Close the file

    // Print the level ed = sf, atan, level, a, b;
    // Find maximum projection (for find peak and valley heights:
    if (proj_pos < 0)
    {
        max_valley_depth = 0;
        min_peak_height = 0;
        return;
    }

```

【図173】

```

break_position = (brk[0] + brk[image.size()-1]) / 2;
// Insert the break
breaklist.add_break(break_position*blast*abvector_height, BREAK_ANGULAR, b
);
}

// Free memory
free_vector(breaklist);
free_vector(a1);
free_vector(a2);
}

// Simple routine used by insert_angular_breaks()
// Checks a value against the current minimum value. If less, it updates
// the min.
// The current minimum
int min;
// Where it's at
double min_at;
// It's angle
int min_theta;
// The test minimum
int test_min;
// Etc....
double test_theta;

if (test_min < 0) // Don't substitute if min_at == -1
return;
if (test_min < min)
min = test_min;
min_at = test_at;
min_theta = test_theta;
}

// Inserts angular breaks in a segment
// Returns number of breaks inserted
// The image
// The breaklist
// The starting break number
// The stopping break number
// The set of thresholds to use
// Projection vector and its size
// The maximum value and what value zero is
// Position of valley
// Maximum height for a valley
// Minimum height for a peak
// Size of the valley walls
// Angles of valley walls
// Used for finding angular projections
// A minimum and where it's located
// A temporary min and where it's located

int proj, proj_alloc;
int max_value_proj;
int valley_pos;
int max_valley_depth;
int min_valley_height;
int i, j;
int test1, test2, test3;
int break_count;
int min_at;
double min_theta;
int test1, test2, test3;
// Zero number of breaks
break_count = 0;

// Allocate the projection vector
proj_alloc = image.height; // This should be more than enough
proj = vector(proj_alloc);

```

[illegible]

【图 1 7 5】

```

// cout << " At theta3 = 4, min = " << min << " in " << "in",
// insert_singular_compute(min, min_at, min_theta, min, min_at,
// theta3+min_degrees);
// Check theta3 = 4
// find_min_value_at_theta(image, blist, start_break-break_count,
// stop_break-break_count, C, theta3+min_degrees, min, min_at);
// cout << " At theta3 = 4, min = " << min << " in " << "in",
// insert_singular_compute(min, min_at, min_theta, min, min_at,
// theta3+min_degrees);
// Check theta3 = 4
// find_min_value_at_theta(image, blist, start_break-break_count,
// stop_break-break_count, C, theta3+min_degrees, min, min_at);
// cout << " At theta3 = 4, min = " << min << " in " << "in",
// insert_singular_compute(min, min_at, min_theta, min, min_at,
// theta3+min_degrees);
// Check theta3 = 4
// find_min_value_at_theta(image, blist, start_break-break_count,
// stop_break-break_count, C, theta3+min_degrees, min, min_at);
// cout << " At theta3 = 4, min = " << min << " in " << "in",
// insert_singular_compute(min, min_at, min_theta, min, min_at,
// theta3+min_degrees);
// Check theta3 = 3
// find_min_value_at_theta(image, blist, start_break-break_count,
// stop_break-break_count, C, theta3+min_degrees, min, min_at);
// cout << " At theta3 = 3, min = " << min << " in " << "in",
// insert_singular_compute(min, min_at, min_theta, min, min_at,
// theta3+min_degrees);
// Check theta3 = 3
// find_min_value_at_theta(image, blist, start_break-break_count,
// stop_break-break_count, C, theta3+min_degrees, min, min_at);
// cout << " At theta3 = 3, min = " << min << " in " << "in",
// insert_singular_compute(min, min_at, min_theta, min, min_at,
// theta3+min_degrees);
// Check theta3 = 4
// find_min_value_at_theta(image, blist, start_break-break_count,
// stop_break-break_count, C, theta3+min_degrees, min, min_at);
// cout << " At theta3 = 4, min = " << min << " in " << "in",
// insert_singular_compute(min, min_at, min_theta, min, min_at,
// theta3+min_degrees);
// Check theta3 = 4
// find_min_value_at_theta(image, blist, start_break-break_count,
// stop_break-break_count, C, theta3+min_degrees, min, min_at);
// cout << " At theta3 = 4, min = " << min << " in " << "in",
// insert_singular_compute(min, min_at, min_theta, min, min_at,
// theta3+min_degrees);
// Check theta3 = 4
// find_min_value_at_theta(image, blist, start_break-break_count,
// stop_break-break_count, C, theta3+min_degrees, min, min_at);
// cout << " At theta3 = 4, min = " << min << " in " << "in",
// insert_singular_compute(min, min_at, min_theta, min, min_at,
// theta3+min_degrees);
// Insert the best break
// cout << " Final at " << min_at << ", theta = " << min_theta+180/3.14159 << ",
// value = " << min << " in " << "in",
// min_at = min_at;
// (if (V) find_min_value_at_theta(image, blist, start_break-break_count,
// stop_break-break_count, C, min_theta, min, min_at);
// }

```



【図176】

```

return b; // Exit (gone to highest level and can't cut)
}
// =====
// CONDUCT METHOD 2 FOLLOWS: (The recombining)
// =====
if (perform_recombining == 0)
    return b;

// Pass each place to recognition (breaklist range = bnum1 to bnum2-b)
// Test to see if MAX_CHARS_PER_CHUNK is high enough
if (bnum2-bnum1-b1 > MAX_CHARS_PER_CHUNK) //
{
    printf("MAX_CHARS_PER_CHUNK in multiLevel.C is too small!\n");
    printf("In this case it needs to be 1000, bnum2-bnum1-b1);
    exit(1);
}

// Load in k[]; the recognition array; 0 = Not Recognized, 1 = Recognized.
// and -1 = end of list
for (i=bnum1; i<bnum2-b; i++) // Loop through each section
{
    bkt = Breaklist_return_break_ptr(blist, i);
    if (bkt->next_recog == -1) // If not previously recognized
    {
        k[i-bnum1] = bkt->next_recog = pass_to_deeper_level(image, blist, i, i+1);
        k[bnum2-bnum1-b] = -1; // End of the Breaklist
    }
    printf("recognize list:\n %i", i);
    while(k[i] != -1)
    {
        if (k[i] == 0)
            printf("N");
        else
            printf("Y");
        i++;
    }
    printf("\n");

    printf("Goodcut Logic: \n"); // If first not recognized
    if (k[0] == 0) // =====// ON LAST EDGE =====
    {
        printf("E RN"); // If RN
        if (k[i] == 0)
        {
            printf("E N MA");
            printf("Trying E N M = ");
            if (pass_to_deeper_level(image, blist, bnum1, bnum2-2)) // Try 0-1
            {
                Breaklist_return_break_ptr(blist, bnum1-1);
                b = -1;
                bkt = Breaklist_return_break_ptr(blist, bnum1);
                bkt->next_recog = 1;
                return b;
            }
            printf("bad\n");
            if (k[i] == -1) // If NRE
            {
                printf("E N M E - Combining E N M E and passing lower\n");
                bkt = Breaklist_return_break_ptr(blist, bnum1-1);
                bkt = Breaklist_return_break_ptr(blist, bnum1);
                bkt->next_recog = -1;
                b = -1;
                b = layer_goodcut(image, blist, bnum1, bnum2-1, thresh_level, 1);
            }
        }
    }
}

```

```

// =====
// Returns the number of breaks added
// =====
int layer_goodcut(
    image* image,
    Breaklist* blist,
    int bnum1,
    int thresh_level,
    int perform_recombining)
{
    int b;
    int bkt;
    int i;
    int i1;
    int i2;
    int bnum2;
    printf("Goodcut Level %d\n", thresh_level);

    // Check to see if width is too long
    bkt = Breaklist_return_break_ptr(blist, bnum1);
    printf("Goodcut level %d, bnum1: %d, bnum2: %d\n", thresh_level, bnum1, bnum2);
    if ((bnum2-bnum1-b1) > MAX_CHARS_PER_CHUNK) //
    {
        printf("Goodcut stopped going up layers because width too small!\n");
        return 0;
    }

    // Insert the breaks
    b = 0;
    while (b == 0) // Loop until break is added
    {
        compute_optimal_threshhold(0, 0, thresh_level, i, i1);
        if (i1 == -1) // If there are no more threshold values to try.
        {
            printf("Max level reached. Returning dom.\n");
            return 0; // Exit and don't go deeper
        }

        // Insert breaks in the image
        b = insert_optimal_breaks(image, blist, bnum1, bnum2, thresh_level);
        printf("Ad breaks inserted at level %d. U1, b, thresh_level, thresh_level\n");

        // Pass each place which is > MAX_LENGTH to deeper levels
        printf("Passing each place that is too long to level.\n");
        for (i=bnum1; i<bnum2-b; i++) // Warning: Deceptive for loop
        {
            bkt = Breaklist_return_break_ptr(blist, i);
            bkt = Breaklist_return_break_ptr(blist, i+1);
            if ((bnum2-bnum1-b1) > MAX_CHARS_PER_CHUNK) // image.size()
            {
                printf("A break was passed to a higher level because too long\n");
                i = layer_goodcut(image, blist, bnum1, bnum2-1, thresh_level, 0);
                if (i == -1) // If didn't bottom out.
                {
                    i = i1; // Check each of the new places
                    b = -1;
                }
            }
        }

        if (b == 0) // If no breaks inserted
        {
            printf("Couldn't find a place to insert a break. Exiting goodcut.\n");
        }
    }
}

```



[ 1 7 8 ]

```

    printf("Combining (N M) and passing deeper\n");
    breaklist_remove_breaklist(bn1);
    breaklist_remove_breaklist(bn2);
    bn2 = breaklist_return_break_ptr(blist, bn);
    bnt = next_recog - 1;
    b == layer_goodcut(image, blist, bn, bn2, thresh_level, 1);
    return b;
}
if (R(bn2) == 0) 44 (R(bn2) == 1) // YNNY
{
    printf("Y N M Y N\n");
    printf("Trying Y (N M) - ");
    if (pass_to_deeper_level(image, blist, bn2, bn4))
    {
        printf("Good\n");
        breaklist_remove_breaklist(bn2);
        bn2 = breaklist_return_break_ptr(blist, bn2);
        bnt = next_recog - 1;
        b == layer_goodcut(image, blist, bn, bn2, thresh_level, 1);
        return b;
    }
    printf("Combining (N M Y) and passing deeper\n");
    breaklist_remove_breaklist(bn1);
    breaklist_remove_breaklist(bn2);
    bn2 = breaklist_return_break_ptr(blist, bn);
    bnt = next_recog - 1;
    b == layer_goodcut(image, blist, bn, bn2, thresh_level, 1);
    return b;
}
printf("WARNING: No pattern matched the goodcut routines\n");
return b;
}

// Master Segmentation Routine
// The input image, a gray image
// Space allocated for a break list
// This will be initialized
breaklist = blist;
int first_break, last_break, // Misc. Used for break numbers
int b, proj_alice, // Projection vector and its size
struct Break *btk;

// Allocate the projection vector
proj_alice = image_size;
proj = vector(proj_alice);

// Find first and last breaks
first_break = 0;
last_break = blist->blist_count-1;
if (last_break != 1)

```

```

    printf("Combining (N M) and passing deeper\n");
    breaklist_remove_breaklist(bn1);
    breaklist_remove_breaklist(bn2);
    bn2 = breaklist_return_break_ptr(blist, bn);
    bnt = next_recog - 1;
    b == layer_goodcut(image, blist, bn, bn2, thresh_level, 1);
    return b;
}
if (R(bn2) == 0) // YNNY
{
    printf("Y N M Y N\n");
    printf("Trying Y (N M) - ");
    if (pass_to_deeper_level(image, blist, bn2, bn4))
    {
        printf("Good\n");
        breaklist_remove_breaklist(bn2);
        bn2 = breaklist_return_break_ptr(blist, bn2);
        bnt = next_recog - 1;
        b == layer_goodcut(image, blist, bn, bn2, thresh_level, 1);
        return b;
    }
    printf("Combining (N M Y) and passing deeper\n");
    breaklist_remove_breaklist(bn1);
    breaklist_remove_breaklist(bn2);
    bn2 = breaklist_return_break_ptr(blist, bn);
    bnt = next_recog - 1;
    b == layer_goodcut(image, blist, bn, bn2, thresh_level, 1);
    return b;
}
printf("WARNING: No pattern matched the goodcut routines\n");
return b;
}

// Master Segmentation Routine
// The input image, a gray image
// Space allocated for a break list
// This will be initialized
breaklist = blist;
int first_break, last_break, // Misc. Used for break numbers
int b, proj_alice, // Projection vector and its size
struct Break *btk;

// Allocate the projection vector
proj_alice = image_size;
proj = vector(proj_alice);

// Find first and last breaks
first_break = 0;
last_break = blist->blist_count-1;
if (last_break != 1)

```



—367—

【图 181】

[illegible]

【図182】

```

1: if(p = fopen("constants.dat", "r")) == NULL)
2:     cerr << "Error in opening constants.dat\n";
3:     exit(1);
4:
5: // *** This should change before the final version. When we know the size
6: // of constants.dat, hardcode it!
7: a = {vector(200)}; // stick the constants somewhere!
8: b = {vector(200)};
9:
10: for (i=0; i<200; i++)
11: {
12:     fscanf(fp, "%f", &a[i], &b[i]);
13:     if ((a[i] < 0) || (b[i] < 0))
14:     {
15:         max_thresh_level = i;
16:         break;
17:     }
18: }
19:
20: if (i==200)
21: {
22:     printf("**** Error: only 200 entries in constants.dat is allowed! Change the size\n");
23:     exit(1);
24: }
25:
26: Const1 = {vector(max_thresh_level)};
27: Const2 = {vector(max_thresh_level)};
28: for (i=0; i<max_thresh_level; i++)
29: {
30:     Const1[i] = a[i];
31:     Const2[i] = b[i];
32: }
33:
34: free_vector(a);
35: free_vector(b);
36:
37: // =====
38: // This must be called before calling compute_rotated_projection
39: // =====
40: void la_initialize_rot()
41: {
42:     int d_index; // Index of degree we're working on
43:     double d_t_d; // The number of degrees, and tangent term
44:     int r; // Looping thru: row on, and degree on
45:     int a,b;
46:
47:     d = ROT_MAX_DEGREE;
48:     r = MAX_CHAR_LENGTH;
49:     Rot = {matrix(a, b)};
50:
51:     for (d = ROT_MIN_RESOLUTION; d<ROT_MAX_DEGREE; d+=ROT_MIN_RESOLUTION)
52:     {
53:         d_index = (int) ((ROT_MIN_RESOLUTION+.5) - 1);
54:         for (a=d+1; a<=ROT_MAX_DEGREE; a++)
55:             for (b=a; b<=ROT_MAX_DEGREE; b++)
56:                 Rot[d_index][a] = (int) ((double) a*b);
57:     }
58: }
59:
60: // ===== PLOTTING ROUTINES =====
61: void la_plot_charlist(
62:     charlist *ch, // The image
63:     struct Charlist *clist, // The character to plot
64:     int x, // The upper left hand corner l.left, .top: to plot
65:     int y)
66: {
67:     int r,c;
68:     for (r=ch->top; r<ch->bot; r++)
69:     {
70:         for (c=ch->curved[r]; c<ch->curved[r+1]; c++)
71:             if (image_pixel[r][c] & 1)
72:                 la_plot_charlist(ch, clist, x, y, r, c);
73:     }
74:     // la_plot_charlist(ch->curved[r+1]-ch->left, y+2);
75:     // la_flush();
76: }
77:
78: void la_plot_charlist(
79:     charlist *ch, // The image
80:     struct Charlist *clist, // The character to plot
81:     int x, // The position
82:     int y)
83: {
84:     struct Character *ch; // The x stating position
85:     int k_start;
86:     k_start = y;
87:     while (k_start < y)
88:     {
89:         if (x + ch->right - ch->left > WINDOW_SIZE_X - 20) // Wrap?
90:             x = k_start;
91:         y = k_start + 15;
92:         la_plot_charlist(ch, x, y);
93:         k_start = ch->right - ch->left;
94:         ch = ch->next; // Get next character
95:     }
96: }
97:
98: // This routine will plot a character in the lower left hand corner
99: // for the purpose of displaying to prompt for recognition!
100: void la_plot_lower_left(
101:     charlist *ch, // The image
102:     struct Character *ch)
103: {
104:     int r,c;
105:     data = ch->bot - ch->top;
106:     for (c=0; c<ch->right-ch->left; c++)
107:         for (r=WINDOW_SIZE_Y-data-15; r<WINDOW_SIZE_Y; r++)
108             la_plot_charlist(ch, r);
109:
110:     for (r=ch->top; r<ch->bot; r++)
111:         for (c=ch->curved[r]; c<ch->curved[r+1]; c++)
112:             if (image_pixel[r][c] & 1)
113:                 la_plot_charlist(ch, r, c);
114:     for (c=0; c<ch->right-ch->left; c++)
115:         // Plot box around it

```

【圖 1 8 3】

```

    }
    for (i=Cursor_Y; i<WINDOW_SIZE_Y; i++)
        ms_plotxy(Cursor_X-10,i);
    else
        for (i=Cursor_Y; i<WINDOW_SIZE_Y; i++)
            ms_plotxy(Cursor_X-11,i);
    Cursor_X += 40; // Minimum of 40 columns over
    ms_flush();

    //***** TEMP RECOGNITION ROUTINE *****
    // Recognized character defined by the ch1 included;
    // The image for unrecognized; 1 otherwise
    struct Image {
        struct Character *ch1; // Left character
        struct Character *ch2; // Right character
    };
    typedef struct Image;
    Image *lower_left_chans(Image, ch1, ch2);
    void lower_left_chans(Image, ch1, ch2);

    //def PART_M_TORNI
    int else_j, else_e,
    char *j;
    else_j = ch1->bot - ch1->top;
    else_e = ch1->right - ch1->left;
    p = matrix(else_j, else_e);
    for (i=0; i<else_j; i++)
        for (j=0; j<else_e; j++)
            p[i][j] = 0;
    for (i=ch1->top; i<ch1->bot; i++)
        for (j=ch1->left; j<ch1->right; j++)
            p[i-ch1->top][j-ch1->left] = Image.pixel(i,j);
    Image *lower_left_chans(Image, else_j, else_e);
    void lower_left_chans(Image, else_j, else_e);

    if ((ch1->right - ch1->left) > MAX_CHAR_LEN)
        return 0;
    printf("Recognised (i-yes, 0-no) ");
    return auto_input_line();

    //***** This routine will determine if a character is too long (and thus exit)
    int is_too_long; // Return 1 if too long, 0 otherwise
    struct Character *ch1; // Left character
    struct Character *ch2; // Right character
    if ((ch1->right - ch1->left) > MAX_CHAR_LEN)
        return 1;
    return 0;

    //***** Character Utility Routines *****
    // This routine will move ch->left, and ch->right so they they are
    // set to the 1st and last 'on' pixel in the image. This is used to

```

```

    ms_plotxy(C, WINDOW_SIZE_Y-1);
    ms_plotxy(C, WINDOW_SIZE_Y-6);
    for (i=WINDOW_SIZE_Y-delta-4; i<WINDOW_SIZE_Y; i++)
        ms_plotxy(0, i);
    ms_plotxy(3-ch->right-ch->left, i);
    ms_flush();

    // This routine will plot a range of characters in the lower left hand corner
    // (for the purposes of displaying to prompt for recognition)
    // Note: They MUST have the same height range!!
    void lower_left_chans(Image, ch1, ch2);
    struct Image {
        struct Character *ch1; // Left character
        struct Character *ch2; // Right character
    };
    typedef struct Image;
    Image *lower_left_chans(Image, ch1, ch2);
    void lower_left_chans(Image, ch1, ch2);

    //def PART_M_TORNI
    int else_j, else_e,
    char *j;
    else_j = ch1->bot - ch1->top;
    else_e = ch1->right - ch1->left;
    p = matrix(else_j, else_e);
    for (i=0; i<else_j; i++)
        for (j=0; j<else_e; j++)
            p[i][j] = 0;
    for (i=ch1->top; i<ch1->bot; i++)
        for (j=ch1->left; j<ch1->right; j++)
            p[i-ch1->top][j-ch1->left] = Image.pixel(i,j);
    Image *lower_left_chans(Image, else_j, else_e);
    void lower_left_chans(Image, else_j, else_e);

    if ((ch1->right - ch1->left) > MAX_CHAR_LEN)
        return 0;
    printf("Recognised (i-yes, 0-no) ");
    return auto_input_line();

    //***** This routine will determine if a character is too long (and thus exit)
    int is_too_long; // Return 1 if too long, 0 otherwise
    struct Character *ch1; // Left character
    struct Character *ch2; // Right character
    if ((ch1->right - ch1->left) > MAX_CHAR_LEN)
        return 1;
    return 0;

    //***** Character Utility Routines *****
    // This routine will move ch->left, and ch->right so they they are
    // set to the 1st and last 'on' pixel in the image. This is used to

```



【図184】

```

int *ang_proj_min, // The min index of ang_proj[] (inclusive) (output)
int *ang_proj_max, // The max index of ang_proj[] (exclusive) (output)
// Note: projection put in proj[]

int r,c; // Looping variables

// Do a standard histogram
// (angle == 0) // Do a standard vpp histogram
{
    *ang_proj_min = range[];
    if (*ang_proj_min < ch->left)
        *ang_proj_min = ch->left;
    *ang_proj_max = ch->right;
    if (*ang_proj_max > ch->right)
        *ang_proj_max = ch->right;

    for (c=ch->left; c<ch->right; c++) // Zero projection array
        ang_proj[c] = 0;
    for (ch=curval[r]; c<ch->curval[r]; c++) // Compute the projection
        if ((image.pixel[r][c] & 1))
            ang_proj[c]++;
    return;
}

// Do a theta is greater than zero
// (angle > 0) // Get the index to use in Rot
angle++;
*ang_proj_min = range[];
if (*ang_proj_min < ch->left)
    *ang_proj_min = ch->left;
*ang_proj_max = range[];
if (*ang_proj_max > ch->right)
    *ang_proj_max = ch->right;
*ang_proj_min = Rot(angle)[ch->bot-ch->top];
for (c=*ang_proj_min; c<*ang_proj_max; c++) // Zero projection array
    ang_proj[c] = 0;
for (c=ch->curval[r]; c<ch->curval[r]; c++) // Compute the projection
    if ((image.pixel[r][c] & 1))
        ang_proj[Rot(angle)[ch->bot-r-1]]++;
return;
}

// Do a theta is less than zero
// (angle < 0) // Get the index to use in Rot
angle--;
*ang_proj_min = range[];
if (*ang_proj_min < ch->left)
    *ang_proj_min = ch->left;
*ang_proj_max = range[];
if (*ang_proj_max > ch->right)
    *ang_proj_max = ch->right;
*ang_proj_min = Rot(angle)[ch->bot-ch->top];
for (c=*ang_proj_min; c<*ang_proj_max; c++) // Zero projection array
    ang_proj[c] = 0;
for (c=ch->curval[r]; c<ch->curval[r]; c++) // Compute the projection
    if ((image.pixel[r][c] & 1))
        ang_proj[Rot(angle)[ch->bot-r-1]]++;
return;
}

// Eliminate white space from the left and right side of a character
// after it is cut so the size won't be too inaccurate.
// Call when pixels the out, chl - left char, chr - right char
void trim_image(char *img)
{
    struct Character *chl, // The left character (only does right side)
    *chr; // The right character (only does left side)

    int r,c; // The new edges of the characters
    int new_left, new_right;
    // Right side of char1. Look for 1st on pixel
    for (c = chl->right-1; c>chl->left; c--)
        for (r = chl->top; r<chl->bot; r++)
            if ((c==chl->curval[r]) & 1) // If pixel is on
                if ((image.pixel[r][c] & 1)) // If pixel is on
                    new_right = c;
    // Break out of the loop
    goto mit_loop;
out_loop:
    // Set right side of character to this value
    chl->right = new_right;
    for (r=chl->top; r<chl->bot; r++)
        if ((chl->curval[r] > new_right))
        {
            chl->curval[r] = new_right;
            if (chl->curval[r] > new_right)
                chl->curval[r] = new_right;
        }
    // Do the left side of chl
    // Look for the 1st on pixel
    for (c=chl->left; c<chl->right; c++)
        for (r=chl->top; r<chl->bot; r++)
            if ((c==chl->curval[r]) & 1) // If pixel is on
                if ((image.pixel[r][c] & 1)) // If pixel is on
                    new_left = c;
    // Break out of the loop
    goto mit_loop;
mit_loop:
    // Set the left side of character to this value
    chl->left = new_left;
    for (r=chl->top; r<chl->bot; r++)
        if ((chl->curval[r] < new_left))
        {
            chl->curval[r] = new_left;
            if (chl->curval[r] < new_left)
                chl->curval[r] = new_left;
        }
}

//***** THE ROTATED PROJECTION ROUTINES *****
void rot_proj(char *img)
{
    struct Character *chl, // The input image
    *chr; // The character to find the projection on
    int angle; // The rotation/ROT_ANGLE_RESOLUTION must be within a
    // valid range
    // The column range to search through
    int range[];
    // The matrix to put the projection into (*MUT*)
    int *ang_proj, // Be previously allocated
}

```

【185】

```

//..... The PVP cache routines .....
// This routine will insert an entry into the PVP cache (A cyclic cache)
// with inputs p1,v1,p2 and outputs loc,angle.
void insert_PVP_cache(insert)
{
    int v1;
    int p1;
    int p2;
    struct Character *ch;
    int min_theta;
    int min_theta;
    // The top of last peak
    // The start of valley
    // The end of valley
    // The top of 2nd peak
    // The end of 2nd peak
    // The final angle
    // Where the minimum is at
    if (DISABLE_PVP_CACHE)
        return;
    int i;
    // Make sure the PVP area doesn't have a cut running through it:
    for (i=ch-stop; i < ch-stop; i++)
        if ((ch-curveval[i] > p1) || (ch-curveval[i] < p2))
            return;
    // If so, don't put on PVP Que
    if (PVP_size)
    {
        PVP_size = 0;
        PVP_size = 1;
        PVP_size = 0;
    }
    else
    {
        if (PVP_size == PVP_SIZE) // List full, swap last out
        {
            // The next line is basically: PVP_size = (PVP_size-1) % CACHE_SIZE
            PVP_size = (PVP_size-1) % PVP_CACHE_SIZE-1;
            PVP_size = PVP_size;
        }
        else
        {
            PVP_size = (PVP_size+1) % PVP_CACHE_SIZE-1;
            PVP_size = PVP_size;
        }
    }
    // Now place the new data at PVP_size
    PVP_p1[PVP_size] = p1;
    PVP_v1[PVP_size] = v1;
    PVP_p2[PVP_size] = p2;
    PVP_ang[PVP_size] = min_theta;
    PVP_angle[PVP_size] = min_theta;
}

// This routine will check PVP against past results and if we've done
// it before, it will just grab the results.
int PVP_cache_hit()
{
    int p1;
    int v1;
    int p2;
    int v2;
    // The top of last peak
    // The start of valley
    // The end of valley
    // The top of 2nd peak
    // The end of 2nd peak
    // The final angle
    // Output: Where the minimum is at
    int min_theta;
    int i;
    if (DISABLE_PVP_CACHE)
        return 0;
    if (PVP_size)
        return 1;
    else
        return 0;
}

```

—373—

[illegible]

【図187】

```

r2 = peak-1;
lr_height = (int) ((float) max_value_vpp * lr_ignore_height * 1);
// how high to ignore the ends

// Find left peak (valley slope is still rising), and above lr_ignore_height
while ((proj[r2] < proj[r1]) || (proj[r2] < proj[r2-1]) ||
       (proj[r2] < proj[r2+1]) || (proj[r2] < lr_height)) {
    if ((--r2) == peak - 3)
        return;
}

// Find right peak (valley slope is falling (from right to left))
while ((proj[r2] < proj[r2-1]) || (proj[r2] < proj[r2+1]) ||
       (proj[r2] < proj[r2+1]) || (proj[r2] < lr_height)) {
    if ((++r2) == peak + 3)
        return;
}

// search this range, and find best minimum
for (lr=1; lr<2; lr++) {
    if (proj[1] < min) // Found a better minimum!
    {
        min = proj[1];
        min_char = 1;
        min_theta = angle;
    }
    else {
        if (fabs(angle) < fabs("min_theta"))
            print("rebreather broken.....\n");
        min_theta = angle;
        min_char = 1;
    }
}

// The find optimum rotated histogram routine
// The first angular break: Attempts to insert break in a character at a given
// threshold. This updates ch-bec and ch-level_out at for both
// characters
int insert_angular_break {
    changed_flag;
    // The character list to add chars to
    struct character *ch;
    int threshold;
    int repeating_char;

    // The start and stop values of the vpp
    static int vpp_min, vpp_max;
    static int max_value_vpp;
    static int max_valley_depth;
    static int min_valley_depth;
    static int min_valley_pos;
    static int min_theta;

    // The maximum height for a valley
    // Minimum height for a peak
    // Position of valley
    // Angles of valley walls (NOT_DPD_RESOLUTION)
    // The Peak-valley-Peak column numbers
    // The location of the minimum
    // The angle of the minimum
}

```

【図188】

```

for (angle = range_t2_min; angle <= range_t2_max; angle++)
    if (find_valley_angle(image, ch, angle, pl, pl,
        min, min_at, min_theta, min_at, min);

    if (vpp_cache_insert(pl, v1, v2, pl, ch, min_theta, min_at);
    ) else {
        printf("Minimum found: angle = %d, at %d = %d\n", min_theta, min_at, min);
        printf("Cache HIT!\n");
    }
    printf("Cache angle = %d, at %d\n", min_theta, min_at);

    *** BEGIN PLOTTING ***
    if (is_visible)
        char string(100);
        int min, max, max_value_proj, tx, ty, unit; // Wrap over to new column!
        if (min > max) max_value_proj = 0;
        if (min < 0) min = 0;
        if (max > 10) max = 10;
        sprintf(string, "new %d", max_value_proj);
        x = string(cursor_x, cursor_y, string);
        cursor_x += 8;
        sprintf(string, "deg %d", min_theta*3);
        x = string(cursor_x, cursor_y, string);
        cursor_x += 8;

        // Load rotated projection into array again
        if (load_projection(image, ch, min_theta, pl, p2, Proj, spin, spinax);
            max_value_proj = Proj[pl];
            for (i = 0; i < Proj[pl]; i++)
                if (Proj[i] > max_value_proj)
                    max_value_proj = Proj[i];
            cursor_y += max_value_proj;
            for (i = 0; i < max_value_proj; i++)
                if (Proj[i] < min)
                    min = Proj[i];
            if (cursor_x < cursor_y + max - min) // Adjust next col loc
                cursor_x = cursor_x + max - min;
            else
                cursor_x = min_at - min;
            // Plot Tickmark
            x = cursor_x;
            y = cursor_y;
            x = plotxy(c-1, z+1); x = plotxy(c+1, z+1);
            x = plotxy(c-1, z+1); x = plotxy(c+1, z+1); x = plotxy(c, z+1);
            x = plotxy(c-1, z+1); x = plotxy(c+1, z+1);
            cursor_y += 10;

            tx = cursor_x; ty = cursor_y; // Plot character (tx, ty)
            if (cursor_x < cursor_y + ch-right - ch-left) // Adjust next col loc
                cursor_x = cursor_x + ch-right - ch-left;
            if (plot_char(image, ch, tx, ty);
                cursor_y += ch-bot - ch-top + 9;
            )
            if (min_theta < 0) // Plot out line
            {
                min = min_theta-1;
                for (i = ch-top-8; i < ch-bot; i++)
                    x = plotxy(tx-bot+1, min+ch-bot-i-1); min_at = ch-left;
            }
            if (min_theta < 0) // Plot out line
            {
                min = min_theta-1;
                for (i = ch-top-8; i < ch-bot; i++)
                    x = plotxy(tx-bot+1, min+ch-bot-i-1); min_at = ch-left;
            }
            if (min_theta < 0) // Plot out line
            {
                for (i = ch-top-8; i < ch-bot; i++)
                    x = plotxy(tx-min+ch-bot-i-1);
            }
        }
    }
}

```

—376—

【図190】

```

c->rec = PASSED_RECOGNITION;
ret_value = l1_product_y3_left(image,clist,c,ich); // Do the y3's
return ret_value;
}
if (c == ch)
{
c->rec = FAILED_RECOGNITION; // Incorporate next character to left
ret_value;
}
// Check for exits
if (c == lch)
{
return 0;
}
if (c->rec == PASSED_RECOGNITION) // If we've passed a y
{
return 0; // Stop searching
}
if (l1_product_y3_left(c,ich)) // If character is too long
{
return 0;
}
}
}
// This routine works with l1_product_comb_right() to check '99's after
// successful recognition, or when a top level is hit.
// Will detect and remove some errors caused by a false successful recognition
// and by not checking adjacent characters when can't split at level 99
int l1_product_start(
    struct Charlist *clist,
    struct Character *ch,
    struct Character *c,
    int ret_value,
    int temp_value)
{
ret_value = temp_value + 2;
c = ch;
while(1)
{
c->prev; // Get the previous character
temp_value--; // At edge
if (c == lch)
{
return ret_value; // Return if not a M
}
return ret_value;
if (c->level_junkat != max_thresh_level) // Return if not '99'
{
return ret_value;
}
if (recognition(image,c,ich))
{
if (combine_chars(clist,c,ich)) // Combine it into one character (in c)
{
ret_value = temp_value;
c = ch;
}
}
}
}
// This routine will start at character ch and search to right free ch
// grouping characters to see if they're recognised. If so they are
// l1_product_comb_right()
// changed image.

Charlist_remove_char(clist, c);
c = c->prev;
}
Charlist_recalculate_locs(clist); // Recalculate location
}
// This routine will start at character ch and search to right free ch
// grouping characters to see if they're recognised. If so they are
// l1_product_comb_right()
// changed image.
int l1_product_y3_left(
    struct Charlist *clist,
    struct Character *ch,
    struct Character *ich)
{
// The character to start from
// Leftmost character to return (exclusive)
// The leftmost character of the range
// The # of chars removed
ret_value = temp_value + 0;
a = ch;
while(1)
{
c->prev; // Get the previous character
temp_value--; // At edge
if (c == lch)
{
return ret_value; // Return if not a M
}
return ret_value;
if (c->level_junkat != max_thresh_level) // Return if not '99'
{
return ret_value;
}
if (recognition(image,c,ich))
{
if (combine_chars(clist,c,ich)) // Combine it into one character (in c)
{
ret_value = temp_value;
c = ch;
}
}
}
}
// This routine will start at character ch and search to left from ch
// grouping characters to see if they're recognised. If so they are
// l1_product_comb_left()
// changed image.
int l1_product_y3_right(
    struct Charlist *clist,
    struct Character *ch,
    struct Character *ich)
{
// The character to start from
// Leftmost character to search (exclusive)
// The leftmost character of the range
// The # of chars removed
ret_value = 0;
c = ch;
while(1)
{
if (recognition(image,c,ich)) // c to ch recognised?
{
if (combine_chars(clist,c,ich)) // Combine it into one character (in c)

```

1

—378—



【図192】

```

}
//
//
void level1(
  cimage* line,          // The line
  struct Charlist *clist) // The charlist
{
  int proj_min, proj_max; // Used to compute size of proj
  int i;

  #ifdef VISIBL
  xs_clr_win();
  int c;
  Cursor1_X = Cursor1_Y = 10;
  line.plot(Cursor1_X, Cursor1_Y); // Plot input image
  Cursor2_Y = Cursor1_Y = line.sizeR + 20; // Final output goes at Cursor2
  Cursor2_X = Cursor1_X;
  Cursor3_Y = Cursor1_Y = line.sizeR + 20;
  Cursor3_X = Cursor1_X;
  for (c=0; c<WINDOW_SIZE_X; c++)
    xs_plotxy(c, Cursor3_Y); // Plot horiz line
  Cursor3_Y = Cursor3_Y + 20;
  Cursor4_Y = Cursor3_Y;
  Cursor4_X = Cursor3_X;
  #endif

  if (line.sizeR > MAX_CHAR_HEIGHT)
  {
    printf("\n*** ERROR ***\n");
    printf("Line size (%d) is greater than max line size allocated in multilevel\n", line.sizeR);
    printf("routines (MAX_CHAR_HEIGHT = %d). Increase MAX_CHAR_HEIGHT.\n", MAX_CHAR_HEIGHT);
    exit(1);
  }

  // Allocate Proj and VPP
  i = (ROT_MAX_RANGE/ROT_DEC_RESOLUTION)-1;
  proj_min = -Rot(1)[line.sizeR];
  proj_max = line.sizeC - proj_min;
  Proj = ivector_ranged(proj_min, proj_max);
  VPP = ivector(line.sizeC);
  VWP_Zero = 1; // Erase the cache

  /* ..... Program goes here ..... */
  goodcut(line, clist);

  #ifdef VISIBL
  // Print final character list
  le_plot_charlist(line, clist, Cursor2_X, Cursor2_Y);
  #endif
  printf("Displaying final level 4 charlist. Enter 0 to continue? ");
  // Print_asc_charlist(clist.clst->first); // ***** DELETE ME *****
  i = auto_input_int();

  // Deallocate
  free_ivector(VPP);
  free_ivector_ranged(Proj, proj_min);
}

```

【図195】

```

/*
 * Filename: mymalloc.h
 * Header file for mymalloc.c
 */
.....

/* Function Definitions for mymalloc.c */
#ifdef CPP
/* Definition for C++ files */
extern "C" {
  int *ivector(int n);
  void free_ivector(int* v);
  float *fvvector(int n);
  void free_fvvector(float* v);
  int *ivector_ranged(int m, int n);
  void free_ivector_ranged(int* v, int m);
  int **imatrix(int r, int c);
  void free_imatrix(int** m, int r);
  char **cmatrix(int r, int c);
  void free_cmatrix(char** m, int r);
}
#else
/* Definition for C files */
/* The header file for mallocs */
int *ivector();
void free_ivector();
int *fvvector();
void free_fvvector();
int *ivector_ranged();
void free_ivector_ranged();
int **imatrix();
void free_imatrix();
char **cmatrix();
void free_cmatrix();
#endif

```

\* \*

【図209】

```

/* This file prints a character to be recognized on the screen by */
/* calling the routine thieu_recognize() */

void toshi_recognize(size_r, size_c, image)
  int size_r; // The number of rows
  int size_c; // The number of columns
  char *image; // The image; Bit 1 is the image
{
  int r,c;

  printf("\nThe character is:\n");
  for (r=0; r<size_r; r++)
  {
    for (c=0; c<size_c; c++)
      if (image[r][c] & 1)
        printf(" "); // On
      else
        printf("."); // Off
      printf("\n");
    }
  printf("***** End of character *****\n\n");
}

```

【図244】

```

colors[x].flags = Dotted | DoGreen | DoBlue;

colors[x].red = the_color;
colors[x].green = the_color;
colors[x].blue = the_color;
the_color += 500;
}
XQueryColors(d, def_colormap, colors, ncolors);

my_cmap = XCreateColormap(d, DefaultRootWindow(d), DefaultVisual(d, scr),
                          AllocAll);
XStoreColors(d, my_cmap, colors, ncolors);
}

```

【図193】

```

/* This program contains common vector and matrix macros */
#include <alloc.h>
#include <stdio.h>
/* Routine for error during memory allocation */
void allocation_error(n)
    char *s;
{
    fprintf(stderr, "%s\n", s);
    exit(1);
}

/* Free an integer ranged vector v of size [m..n-1] */
void free_vector_ranged(v, m)
    int *v;
    int m;
{
    free((int *) (v+m));
}

/* Allocate an integer matrix of size [0..r-1][0..c-1] */
int **matrix(r, c)
    int r, c;
{
    int **m;
    int i;
    m = (int **) malloc((unsigned) r * sizeof(int));
    if (!m) allocation_error("in matrix()");
    for (i=0; i<r; i++) {
        m[i] = (int *) malloc((unsigned) c * sizeof(int));
        if (!m[i]) allocation_error("in matrix()");
    }
    return m;
}

/* Free an integer matrix m of size [0..r-1][0..c-1] */
void free_matrix(m, r)
    int **m;
    int r;
{
    int i;
    for (i=0; i<r; i++)
        free((int *) m[i]);
    free((int **) m);
}

/* Allocate an char matrix of size [0..r-1][0..c-1] */
char **matrix(r, c)
    int r, c;
{
    char **m;
    int i;
    m = (char **) malloc((unsigned) r * sizeof(char));
    if (!m) allocation_error("in matrix()");
    for (i=0; i<r; i++) {
        m[i] = (char *) malloc((unsigned) c * sizeof(char));
        if (!m[i]) allocation_error("in matrix()");
    }
    return m;
}

/* Free an character matrix m of size [0..r-1][0..c-1] */
void free_matrix(m, r)
    char **m;
    int r;
{
    int i;
    for (i=0; i<r; i++)
        free((char *) m[i]);
}

/* This program contains common vector and matrix macros */
#include <alloc.h>
#include <stdio.h>
/* Routine for error during memory allocation */
void allocation_error(n)
    char *s;
{
    fprintf(stderr, "%s\n", s);
    exit(1);
}

/* Free an integer ranged vector v of size [m..n-1] */
void free_vector_ranged(v, m)
    int *v;
    int m;
{
    free((int *) (v+m));
}

/* Allocate an integer matrix of size [0..r-1][0..c-1] */
int **matrix(r, c)
    int r, c;
{
    int **m;
    int i;
    m = (int **) malloc((unsigned) r * sizeof(int));
    if (!m) allocation_error("in matrix()");
    for (i=0; i<r; i++) {
        m[i] = (int *) malloc((unsigned) c * sizeof(int));
        if (!m[i]) allocation_error("in matrix()");
    }
    return m;
}

/* Free an integer matrix m of size [0..r-1][0..c-1] */
void free_matrix(m, r)
    int **m;
    int r;
{
    int i;
    for (i=0; i<r; i++)
        free((int *) m[i]);
    free((int **) m);
}

/* Allocate an char matrix of size [0..r-1][0..c-1] */
char **matrix(r, c)
    int r, c;
{
    char **m;
    int i;
    m = (char **) malloc((unsigned) r * sizeof(char));
    if (!m) allocation_error("in matrix()");
    for (i=0; i<r; i++) {
        m[i] = (char *) malloc((unsigned) c * sizeof(char));
        if (!m[i]) allocation_error("in matrix()");
    }
    return m;
}

/* Free an character matrix m of size [0..r-1][0..c-1] */
void free_matrix(m, r)
    char **m;
    int r;
{
    int i;
    for (i=0; i<r; i++)
        free((char *) m[i]);
}

```

【圖 196】

—381—

【図197】

```

startingx = t_cursorx + 1; // Initialize cursor
startingy = t_cursory + 1; // Facing left
t_start = left_boundary;
t_stop = right_boundary;
not_finished = 1;
while(not_finished) // While we haven't returned to starting
{
    // Adjust cursor for this new position
    line_pixel(t_cursorx)(t_cursory) = line_pixel(t_cursorx)(t_cursory) + 2;
    if (t_cursorx < curvel(t_cursorx)) // Set second bit to 1 (outlined)
    {
        curvel(t_cursorx) = t_cursorx; // Move break
        if (t_cursorx < left) // Update left
        {
            left = t_cursorx;
        }
        if (t_cursorx < top) // Update top and bot
        {
            top = t_cursorx;
            bot = t_cursorx;
        }
        if (t_cursorx > curvel(t_cursorx)-1) // (Same for right side)
        {
            curvel(t_cursorx) = t_cursorx; // Move, we need to update and
            if (t_cursorx > right) // and make because the bit
            {
                right = t_cursorx; // (Same for left side)
            }
        }
    }
    // Check to see if we've entirely circled the image
    if ((t_cursorx == startingx) && (t_cursory == startingy))
    {
        // We've returned to the starting point
        i = t_cursorx; // Remember direction
        t_cursory = up; // Check the space above
        if ((t_infront(i) & 7) != 1) // If space above is on, and hasn't
        {
            t_cursorx = LEFT; // been outlined, don't exit
        }
        if ((t_infront(i) & 7) != 1) // Likewise for the left space,
        {
            not_finished = 0; // we're finished, exit the while routine
        }
        break;
    }
    t_cursorx = 1; // Restore direction and continue
    // Move mouse to next position along the edge
    t_cursory = left; // Check area to left of mouse
    if ((t_infront(i) & 5) == 1) // If something there
    {
        t_cursory = forward(); // Move to left
        continue;
    }
    t_cursory = right; // Check area in front of mouse
    if ((t_infront(i) & 5) == 1) // If something there
    {
        t_cursory = forward(); // Move forward
        continue;
    }
    t_cursory = left; // Check area to right of mouse
    if ((t_infront(i) & 5) == 1) // If something there
    {
        t_cursory = forward(); // Move to right
        continue;
    }
    t_cursory = forward(); // Add loop again
    continue;
}

```

```

}
// This routine will determine the extent of where the mouse is facing
// It returns 0 if it is out of range
int t_infront(image, image)
{
    int oldx, oldy; // Stores the old direction
    int p; // Put to return
    oldx = t_cursorx; // Remember this position
    oldy = t_cursory;
    // Move into the new space
    if (t_infront > image.size())
    {
        t_cursory = oldx; // Restore cursor position
        t_cursory = oldy; // It too high or low,
        return 0;
    }
    if (t_cursory < t_start)
    {
        t_cursory = t_start;
    }
    if (t_cursory > t_stop)
    {
        t_cursory = t_stop;
    }
    // Restore cursor position
    return 0; // It to the left or right of the boarder
}
p = image.pixel(t_cursorx)(t_cursory);
t_cursory = oldx;
t_cursory = oldy;
return p;
}

// Main outlining routine
void outline(image, image)
{
    // Loop of the line
    // The character line
    // The left boundary of the area to outline
    // The right boundary of the area to outline
    // The row and column of an 'on' pixel
    // On the edge
    struct Character *new_char;
    int top, bot, left, right; // Used to be put in the new character
    int *curvel, *curvel; // Pointers to curvel and curvel of new char.
    int startingx, startingy; // The starting position
    int not_finished; // Set to 1 when finished looping
    int i;
    // Set up convenient pointers to the new character we're inserting
    new_char = malloc(sizeof(Character));
    curvel = new_char->curvel;
    curvel = new_char->curvel;
    // Set in address of curves 1 & 3
    // Initialize variables of new outlined character
    for (i=0; i<sizeof(Character); i++) // Initialize curve arrays
    {
        curvel[i] = 0;
        curvel[i] = 0;
    }
    top = bot = 0; // Max height
    left = right = 0; // Max left and right entrance
    // Initialize turtle stuff
}

```

【図198】

```

t_turn_right(); // Check area to behind of mouse
if ((t_in_front(line) & 3) == 1) // If something there
{
    t_move_forward(); // Move behind mouse
    continue; // And loop again
}
not_finished = 0; // Else it's a one-pixel image. so exit
line.pixel[t_cursor][t_cursor] = line.pixel[t_cursor][t_cursor] + 4;
// Make this spot!
}

// Load the info into the Charlist
new_char->step = top;
new_char->bot = bot+1; // (Make it exclusive)
new_char->left = left;
new_char->right = right+1; // (Make it exclusive)
new_char->type = CHARACTER_PURE_OUTLINED;

Charlist_place_new_char(clist); // Insert the new character

// Mark the character as removed from the list
for (r=0; r<line.size; r++)
for (c=curve1[r]; c < curve2[r]; c++)
    line.pixel[r][c] = line.pixel[r][c] + 4;
}

////////////////////////////////////
//// Main routine to call to add breaks to the image //
////////////////////////////////////
void do_outlining() // Returns number of characters added
{
    changed line; // Image of the line
    struct Charlist* clist; // The character list
    int start_col; // The range to search through (inclusive)
    int stop_col; // (exclusive)

    int r,c; // The row and column we're on

    for (c=stop_col-1; c>=start_col; c--) // Search through column range
    for (r=line.size-1; r>=0; r--) // Scan from bottom to top
        if (((line.pixel[r][c]) & 3) == 1) // On pixel that hasn't been outlined
        {
            line.pixel[r][c] = 123;
            print_outlining(line, c);
            extract_outline_char(line, clist, start_col, stop_col, r, c);
            // Outline the character
        }
}

```

【図203】

```

/*
 * The routines in this file determine if a character is recognized or not
 * (for segmentation)
 */
.....
#include "x.h" // Include Rick's graphics routines */
#include "autoinput.h"

extern int WINDOW_SIZE_Y;

int recognised(a, size_x, size_c) /* Return a 0 for not recognized, 1 for
recognized */
{
    int **a; // The character matrix a[row][col] */
    int size_x; // The number of rows in a */
    int size_c; // The number of columns in a */

    int r,c; // The row and column of the image */
    int dummy; // A dummy variable */

#ifdef VISIBL
/* This routine will print the char at the screen's bottom left corner */
for (r=0; r<size_x; r++) // Loop through the rows */
for (c=0; c<size_c; c++) // Loop through the cols */
    if ((r>0) && (c<size_x) && (c>0) && (c<size_c)) // If the pixel is on */
        if ((a[r][c]) & 3) // If the pixel is on */
            xm_plotxy(c+20, r + WINDOW_SIZE_Y - size_x + 20);
        else
            xm_unplotxy(c+20, r + WINDOW_SIZE_Y - size_x + 20);
        else
            xm_unplotxy(c+20, r + WINDOW_SIZE_Y - size_x + 20);

xm_flush(); // Flush the display buffer (to make sure
the character gets printed on the screen */
#endif
printf("Character displayed. Enter return value (1 = good, 0 = bad) ");
dummy = auto_input_int();
return dummy;
}

```

[illegible]

[図200]

```

loc **hgram; // Array of histograms; [num][row]
loc **startcol; // Array of starting text positions
loc **stopcol; // Array of stopping text positions
int **cursor; // The cursor for the start and stop texts
int **size; // The size of the histogram
int **end_of_list; // Include this element in the next line (0=no, 1=yes)
int top_div; // Loop termination flag
int range1, range2; // The division with the topmost text
int crange1, crange2; // The text range in the adjacent division (row's)
int min_line_height; // The text range in the current division (row's)
int min_line_height; // The minimum line height
int reject; // Line rejection flag
int i, j, a, b, r, c; // Nice handy variable

stopcol = (vector<INDIVISIONS>);
hgram = (vector<INDIVISIONS>);
hgram = matrix<INDIVISIONS, page_size>; // This is overkill!
startcol = matrix<INDIVISIONS, page_size>; // This is overkill!
cursor = (vector<INDIVISIONS>);
size = (vector<INDIVISIONS>);
end_of_list = (vector<INDIVISIONS>);

min_line_height = (int) ((float) page_size) * PDEF_DEF_MIN_BLOCK_SIZE;

////////// Step 1: Define columns and find histograms //////////
// Find range of column
startcol[0] = PAGE_SIZE;
for (i=0; i<INDIVISIONS-1; i++)
{
    e = PAGE_SIZE * PAGE_SIZE * (i+1) / INDIVISIONS;
    min_line_height = min(min_line_height, e);
    stopcol[i] = e + OVERLAY;
}
// Find histograms
for (i=0; i<INDIVISIONS; i++) // For each division
{
    make_partial_histo(page, startcol[i], stopcol[i], hgram[i]);
}

////////// Step 2: Make text area lines //////////
for (i=0; i<INDIVISIONS; i++)
{
    size[i] = 0;
    for (j=PAGE_SIZE-1; j>PAGE_SIZE; j--) // Initialize
    {
        if (hgram[i][j] == 0) // If starting text
            startcol[i][j] = j-1;
        if (hgram[i][j] != 0) // If stopping text
            stopcol[i][j] = j;
    }
    size[i] = j;
}

////////// Step 3: Find the lines //////////
for (i=0; i<INDIVISIONS; i++) // set all pointers to the top of the list
{
    current[i] = 0;
    end_of_list[i] = 0;
}
while(end_of_list == 0)

```

```

for (i=0; i<PAGE_SIZE; i++) // Zero before PAGE_SIZE
{
    for (j=PAGE_SIZE; j<PAGE_SIZE; j++) // Histogram between A1 and B2
    {
        a[i] = 0;
        for (c=a1; c<=a2; c++)
        {
            if (page_size[c] != 0)
            {
                a[i]++;
                c = n;
            }
        }
        for (i=PAGE_SIZE; i<page_size; i++) // Zero after PAGE_SIZE
        {
            a[i] = 0;
        }
    }
}

// Given a range (a1,a2) and (b1,b2) will compute fraction of union/intersection
// Required: a1 <= a2 and b1 <= b2, and the ranges overlap.
// If not overlap, will give a negative number
float overlap(
    int a1, // Range 1
    int b1, // Range 2
    int b2)
{
    int u1, u2; // The intersection range
    int u1, u2; // The union range

    if (a1 < b1) {
        u1 = b1;
        u2 = a1;
    } else {
        u1 = a1;
        u2 = b1;
    }

    if (a2 < b2) {
        u1 = a2;
        u2 = b2;
    } else {
        u1 = b2;
        u2 = a2;
    }

    if (u2 == u1) // Only you can prevent floating point exceptions!
        return 0.0;
    return ((float) (u2-u1) / (float) (u2-u1));
}

// This routine will segment an entire page given the page
// It will call itself recursively for each page
// Uses PAGE_SIZE... Note: PAGE_SIZE must NOT be zero, and PAGE_SIZE must NOT
// be larger than size.
// void segment_page(
//     char* page, // A character list (memory allocated only)
//     struct CharList *clist, // The lines, one at a time
//     int *startcol, // The start of a column (inclusive)
//     int *stopcol, // The stop of a column (exclusive)

```

—386—

[illegible]





【☒204】

—388—

【図205】

```

// Get next character and move cursor over
last_right_edge = chr->right;

// Get next cursor
chr = chr->next;

// Update the cursor (slide to right)
if (chr)
    seep_cursor = new_cursor + 1;
else
    seep_cursor = new_cursor + chr->left - last_right_edge + 2;
}
ms_flush();

seep_cursor += max_c_height/SCALE + 10; // Reposition the cursor for next line
seep_cursor = LEFT_EDGE;

// Print the character list
printf("Enter 0 to continue? (not auto input)? ");
scanf("%d", &t);
}

// Check if we need to scroll... Enter 0 to continue?
scanf("%d", &t);
if (t)
    seep_bot_row_printed = seep_bot_row_printed;
else
    seep_bot_row_printed = seep_bot_row_printed;
seep_cursor = LEFT_EDGE;
seep_cursor = WINDOW_SIZE_Y/2 + 10;

// Print the character list
chr = char->first;
while (chr != 0)
{
    // Check if need to scroll one line
    if (seep_cursor + chr->right - chr->left + 10 > WINDOW_SIZE_X)
    {
        // Scale
        seep_cursor = max_c_height/SCALE + 10; // Move down
        seep_cursor = LEFT_EDGE + 10; // Indent
        y+=10;
        seep_cursor = seep_cursor; // The y
        seep_cursor = seep_cursor; // Initialize cursor
        seep_cursor = seep_cursor; // The y
        for (t=0; t<limit.size(); t++)
        {
            if (t >= chr->stop) break;
            if (t < y)
                y = t;
            if (t < y)
                y = t;
            for (c=chr->left; c < chr->right; c++)
            {
                if ((t >= chr->curved[t]) && t < chr->curved[t+1])
                    seep_cursor = t;
                t++;
            }
            seep_cursor = t;
            t = seep_cursor;
        }
        y++;
    }
    // Draw box
    seep_cursor = seep_cursor;
    for (t=seep_cursor-1; t<seep_cursor+1; t++)
    {
        seep_cursor = t;
        seep_cursor = t;
        for (y=y-1; y<=y+1; y++)
        {
            seep_cursor = seep_cursor-1;
            seep_cursor = seep_cursor;
        }
    }
}

```

【図206】

```

/* This program will test the charlist */
#include "charlist.h"

void print_charlist(cclist)
struct Charlist *cclist;
{
    struct Character *ch;

    printf("Charlist:\n");
    ch = cclist->first;

    while(ch != 0) {
        printf("Loc: %d, Chars (-..-) = ('', ch->loc/8);",
            (ch->prev == 0) ? 0 : (ch->prev->stop));
        if (ch->prev == 0)
            printf("****");
        else
            printf("%d", ch->prev->stop);
        printf("%d", ch->stop);
        if (ch->next == 0)
            printf("****");
        else
            printf("%d", ch->next->stop);
        printf("\n");
        ch = ch->next;
    }
    printf("End of charlist\n");
}

main()
{
    struct Character *ch;
    struct Charlist cclist;
    int a[10]; /* Dummy vector for height */
    int i, p;

    Charlist_initialize(&cclist, 10, 10);

    for (i=0; i<1; i++) {
        printf("Position of new Char %d? ", i);
        scanf("%d", &p);
        ch = &cclist.chr[cclist.size];
        ch->stop = i;
        ch->bot = i+1;
        ch->curve1[i] = ch->curve2[i] = p;
        Charlist_place_new_char(&cclist);
        print_charlist(&cclist);
    }

    while(1) {
        printf("Replace which char? ");
        scanf("%d", &i);

        ch = cclist.first;
        while(ch != 0) {
            if (ch->stop == i)
                break;
            ch = ch->next;
        }
        if (ch == 0) {
            printf("Character not found.\n");
        } else {
            printf("New position? ");
        }
    }
}

```

【図220】

```

/*
 * Filename: autoinput.h
 * Header file for autoinput.c
 */

/* Function Definitions for mymalloc.c */
#ifdef CPP
/* Definition for C++ files */
extern "C" {
    void auto_initialize();
    int auto_input_int();
    void auto_terminate();
}
#else
/* Definition for C files */
void auto_initialize();
int auto_input_int();
void auto_terminate();
#endif

```

【図223】

```

/*-----*/
void ZeroVpp_terminate(nzlist)
struct MonZeroVppList *nzlist; /* The list */
{
    free_vector(nzlist->list); /* Deallocate memory */
    nzlist->allocated = 0; /* Set so won't be accidentally used */
    nzlist->size = 0;
}

/*-----*/
void ZeroVpp_add(nzlist, a, b)
struct MonZeroVppList *nzlist; /* The list */
int a, b; /* Start and stop of the zeroVpp */
{
    if (nzlist->size+2 > nzlist->allocated) {
        printf("MonZeroVppList FULL!\n");
        exit(1);
    }
    nzlist->list[(nzlist->size++)] = a;
    nzlist->list[(nzlist->size++)] = b;
}

```

【図207】

```

// filename: test_mult.c
// This file is used to test and develop multilevel2.c
// =====
//
#include <stdio.h>
#include "images.h"
#include "ymalloc.h"
#include "xs.h"
#include "charlist.h"

extern void l4_initialize();
void level4(image_t line, struct Charlist *clist);

void xs_init()
{
    struct Charlist clist;
    struct Character *ch;
    image_t image;
    int i, j;

    l4_initialize();
    Charlist_initialize(&clist, 10, 380); // 10 chars, 100max height
    ch = &clist.chr[clist.ele]; // Set new_char to the last character

    // =====
    // Load the image
    //
    image_fill(1, 10, 10);
    image.pixel[6][6] = 0;
    image.pixel[7][6] = 0;
    image.pixel[8][6] = 0;
    image.pixel[6][7] = 0;
    image.pixel[7][7] = 0;
    image.pixel[8][7] = 0;
    image.pixel[6][8] = 0;
    image.pixel[7][8] = 0;
    image.pixel[8][8] = 0;
    image.pixel[6][9] = 0;
    image.pixel[7][9] = 0;
    image.pixel[8][9] = 0;
    image.pixel[2][10] = 0;
    image.pixel[3][10] = 0;
    image.pixel[4][10] = 0;
    image.pixel[5][10] = 0;
    image.pixel[6][10] = 0;
    image.pixel[7][10] = 0;
    image.pixel[8][10] = 0;
    image.pixel[2][11] = 0;
    image.pixel[3][11] = 0;
    image.pixel[4][11] = 0;
    image.pixel[5][11] = 0;
    image.pixel[6][11] = 0;
    image.pixel[7][11] = 0;
    image.pixel[8][11] = 0;
    image.pixel[2][12] = 0;
    image.pixel[3][12] = 0;
    image.pixel[4][12] = 0;
    image.pixel[5][12] = 0;
    image.pixel[6][12] = 0;
    image.pixel[7][12] = 0;
    image.pixel[8][12] = 0;
    image.pixel[2][13] = 0;
    image.pixel[3][13] = 0;
    image.pixel[4][13] = 0;
    image.pixel[5][13] = 0;
    image.pixel[6][13] = 0;
    image.pixel[7][13] = 0;
    image.pixel[8][13] = 0;

    image.pixel[0][14] = 0;
    for (i=1; i<9; i++)
        image.pixel[i][20] = 0;
    for (i=0; i<10; i++)
        image.pixel[i][21] = 0;
    for (i=1; i<9; i++)
        image.pixel[i][22] = 0;
    for (i=2; i<8; i++)
        for (j=2; j<8; j++)
            image.pixel[j][1] = 0;
    for (i=2; i<8; i++)
        for (j=1; j<2; j++)
            image.pixel[j][1] = 0;
    for (i=2; i<8; i++)
        for (j=7; j<8; j++)
            image.pixel[j][1] = 0;

    // Load the character
    ch->stop = 0;
    ch->bot = 10;
    ch->left = 0;
    ch->right = 33;
    for (i=ch->stop; i<ch->bot; i++) {
        ch->curve1[i] = ch->left;
        ch->curve2[i] = ch->right;
    }

    Charlist_place_new_char(&clist);
    level4(image, &clist);

    printf("**** End of Program ****\n");
}

```

【図235】

```

/*
 * filename: ymalloc.h
 * Header file for ymalloc.c
 *
 * =====
 */

/* Function Definitions for ymalloc.c */
#ifndef CPP
/* Definition for C++ files */
extern "C" {
    int *ivector(int n);
    void free_ivector(int* v);
    float *fvector(int n);
    void free_fvector(float* v);
    int *ivector_ranged(int n, int m);
    void free_ivector_ranged(int* v, int n);
    int **imatrix(int r, int c);
    void free_imatrix(int** m, int r);
    char **cmatrix(int r, int c);
    void free_cmatrix(char** m, int r);
}
#else
/* Definition for C files */
/* The header file for mallocs */
int *ivector();
void free_ivector();
int *fvector();
void free_fvector();
int *ivector_ranged();
void free_ivector_ranged();
int **imatrix();
void free_imatrix();
char **cmatrix();
void free_cmatrix();
#endif

```

【図240】

```

/* This file prints a character to be recognized on the screen by */
/* calling the routine thier_recognize() */

char thier_recognize(size_r, size_c, image)
    int size_r; /* The number of rows */
    int size_c; /* The number of columns */
    char **image; /* The image. Bit 1 is the image */
{
    int r, c;
    char input_string[100];

    printf("\nthe character is:\n");
    for (r=0; r<size_r; r++)
    {
        for (c=0; c<size_c; c++)
            if (image[r][c] & 1)
                printf(" "); /* On */
            else
                printf("."); /* Off */
        printf("\n");
    }
    printf("***** End of character *****\n\n");
    printf("Enter Character? ");
    scanf("%s", input_string);
    return input_string[strlen(input_string)-1];
}

```

【図208】

```

//
// Filename: test_mult.C
// This file is used to test and develop multilevel2.C
//
//
//include <stdio.h>
//include <string.h>
//include "image.h"
//include "xmalloc.h"
//include "xs.h"
//include "charlist.h"

extern void l4_initialize();
void level4(image* limg, struct Charlist *clist);

void xs_init()
{
    struct Charlist clist;
    struct Character *ch;
    cimage image;
    int i,j;

    l4_initialize();

    Charlist_initialize(&clist, 500, 300); // 10 chars, 100-max height

    char filename1[100];
    char filename2[100];
    printf("Input filename? /d3/cif/");
    scanf("%s", filename1);
    strcpy(filename2, "/d3/cif/");
    strcat(filename2, filename1);
    image.readtif(filename2);
    image.clipspage();

    ch = &clist.chr(clist.size); // set new_char to the last character

    // Load the character
    ch->top = 0;
    ch->bot = image.size1;
    ch->left = 0;
    ch->right = image.size2;
    for (i=ch->top; i<ch->bot; i++) {
        ch->curve1[i] = ch->left;
        ch->curve2[i] = ch->right;
    }

    Charlist_place_new_char(&clist);

    level4(image, &clist);

    printf("**** End of Program ****\n");
}

```

【図228】

```

c1 = get2bytes(f, fp);
c2 = get2bytes(f, fp);
return c1 + c2*4096;
}

// Plot image using Flash Graphics at (x0, y0) with color
void cimage::plot(int x, int y)
{
    int px,py; // Pixel coords on the screen (upper left)
    char color;

    for (py=0; py<size1; py++)
        for (px=0; px<size2; px++)
            {
                color = pixel(py)[px];
                if (color & 1)
                    xs_plotxy(x+px, y+py); // Draw it
            }
};

// Plot image with a box around it (with color 1)
void cimage::plotbox(int x, int y)
{
    int px,py;

    for (px=0; px<size2; px++) { // Draw horizontal lines
        if ((x+px)%42 == 0)
            xs_plotxy(x+px, y);
        if ((x+px+size2-1)%42 == 0)
            xs_plotxy(x+px, y+size1);
    }
    for (py=1; py<size1; py++) {
        if ((x+py)%42 == 0)
            xs_plotxy(x, y+py);
        if ((x+size2-1+y)%42 == 0)
            xs_plotxy(x+size2-1, y+py);
    }
    plot(x+1, y+1); // Plot the image
}

```

【☒ 2 1 0】

—393—





【図212】

```

/* Routine: xa_redisplay()
Description: Redisplay the 'stuff' that was on the window
Return: Nothing
Author: Rick Lee
*/
xa_redisplay()
{
    /* Routine: xa_redisplay(canvas), xWindow(canvas), 0.0, 0.0, TRUE,
    /* Routine: xa_draw_box(x,y,width,height)
Description: Draw a box on screen
Return: Nothing
Author: Rick Lee
*/
    xa_draw_box(x,y,width,height)
    int w;
    int y;
    int width;
    int height;
    {
        XDrawRectangle(XDisplay(canvas),xWindow(canvas),line_gc,
            x,y,width,height);
    }
}

/* Routine: xa_set_color
Description: Set the current color
Return: Nothing
Author: Rick Lee
*/
xa_set_color(color)
int color;
{
    gc.setForeground = color;
    XSetForeground(XDisplay(canvas),line_gc,color);
}

```

```

/* Routine: xa_redisplay()
Description: Redisplay the 'stuff' that was on the window
Return: Nothing
Author: Rick Lee
*/
xa_redisplay()
{
    /* Routine: xa_draw_bitmap(x,y,width,height,buf)
Description: Draw the given bitmap
Return: Nothing
Author: Rick Lee
*/
    xa_draw_bitmap(x,y,width,height,buf)
    {
        bitmap bitmap_pix; /* The bitmap for display */
        /* Reverse the bit ordering */
        uc_reverse(width,height,buf);
        /* First create bitmap */
        bitmap = XCreateBitmapFromData(XDisplay(canvas),
            RootWindowOfScreen(XScreen(canvas)),
            buf,width,height);
        /* Create the pixmap area */
        pix = XCreatePixmap(XDisplay(canvas),RootWindowOfScreen(XScreen(canvas)),
            width,height,depth);
        /* Now copy the data to pixmap */
        XCopyPlane(XDisplay(canvas),bitmap,pix,line_gc,0,0,
            width,height,0,0);
        /* Now copy the pixmap to drawing area */
        XCopyArea(XDisplay(canvas),pix,XWindow(canvas),line_gc,0,0,
            width,height,x,y);
        /* Now free the pixmap */
        XFreePixmap(XDisplay(canvas),bitmap);
        /* Now free the bitmap */
        XFreeBitmap(XDisplay(canvas),pix);
        /* Now flush the queue to X-server */
        XFlush(XDisplay(canvas));
    }
}

/* Routine: xa_get_win()

```

【図213】

```

/*
Routine: xs_win_dim
Description: Get the size of the window
Return: Nothing
Author: Rick Lee
*/
xs_win_dim(win_width, win_height)
int *width;
int *height;
{
    Arg args[2];
    Dimension win_width, win_height;
    XtSetArg(args[0], XtWidth, &win_width);
    XtSetArg(args[1], XtHeight, &win_height);
    XtGetValues(canvas, args, 2);
    *width = win_width;
    *height = win_height;
}

/*
xs_set_cmap(w)
Widget w;
{
    int ncolors;
    XColor colors[256];
    Colormap my_cmap;
    int count = 1;

    Display *d;
    int scr;
    Colormap def_cmap;
    XColor color;
    int x;
    int the_color = 3000;

    d = XtDisplay(w);
    scr = DefaultScreen(d);
    ncolors = DisplayCells(d, scr);
    def_cmap = DefaultColormap(d, scr);
    for(x = 0; x < ncolors; x++)
    {
        colors[x].pixel = x;
        colors[x].flags = DoRed | DoGreen | DoBlue;

        colors[x].red = the_color;
        colors[x].green = the_color;
        colors[x].blue = the_color;
        the_color -= 500;
    }
    XQueryColors(d, def_cmap, colors, ncolors);
    my_cmap = XCreateColormap(d, DefaultRootWindow(d), DefaultVisual(d, scr),
                             AllocAll);
    XStoreColors(d, my_cmap, colors, ncolors);
}
*/

```

【図215】

```

/* If your routines are in C++ then CPP must be defined in make file */
#ifdef CPP
extern "C" void xs_init();
extern "C" void xs_flush();
extern "C" void xs_plotxy(int x, int y);
extern "C" void xs_unplotxy(int x, int y);
extern "C" void xs_redisplay();
extern "C" void xs_dls_bitmap(int x, int y, int width, int height, char *buf);
extern "C" void xs_dls_win();
extern "C" void xs_draw_box(int x, int y, int width, int height);
extern "C" void xs_string(int x, int y, char *str);
extern "C" void xs_setcolor(int color);
extern "C" void xs_win_dim(int *width, int *height);
#endif

```

【図215】

```

int n,i;
int no_set_win_bg; /* = 1 to not set window size */
int no_set_win_bg; /* = 1 to not set window background */

//get CIP
/* Call the routine to setup C++ interface, take out if no C++ interface */
main()
{
    /* Checks for geometry and background options in command line */
    no_set_win_bg = 0;
    for (i=0; i<argc; i++) {
        if (strcmp(argv[i], "-geometry") == 0)
            no_set_win_bg = 1;
        if (strcmp(argv[i], "-background") == 0)
            no_set_win_bg = 1;
        if (strcmp(argv[i], "-width") == 0)
            no_set_win_bg = 1;
        if (strcmp(argv[i], "-height") == 0)
            no_set_win_bg = 1;
        if (strcmp(argv[i], "-bg") == 0)
            no_set_win_bg = 1;
    }

    /* Init the Xt function */
    parent = XtInitialize(argv[0], "Xmap", 0, NULL, 0, argc, argv);

    /* XtGetApplicationResources(parent, 0, NULL, 0, argv, n); */

    /* XtSetBackground(Display(parent), line_bg);
    /* WhiteOut(Display(parent));
    /* DefaultScreen(Display(parent));

    /* Create drawing area widget */
    canvas = XtCreateManagedWidget("canvas", XDrawnWidgetClass,
    parent, NULL, 0);

    /* Set the GC for display routines */
    line_gc = XCreateGC(Display(canvas),
    0, 0, 0);

    /* Now add callbacks and event handler */
    XtAddCallback(canvas, XDrawnCallback, xt_redisplay, NULL);

    /* Set the default screen size */
    if (no_set_win_bg == 0) {
        XSetArg(args[0], XNWidth, DEFAULT_WINDOW_SIZE_WIDTH);
        XSetArg(args[1], XNHeight, DEFAULT_WINDOW_SIZE_HEIGHT);
        XtSetValues(canvas, args, 2);
    }

    /* Display the widget and enter event loop */
    XtRealizeWidget(parent);
    XtMainLoop();
}

```

```

// FILE: xt.C
// COMMENTS:
// This file contains interface to X-display routines.
//
// main()
// xt_redisplay(w, data, call_data);
//
// Usage:
// Your initialize routine must be xt_init().
//
// #define DEFAULT_WINDOW_SIZE_HEIGHT 400
// #define DEFAULT_WINDOW_SIZE_WIDTH 1130
//
// -----Includes-----
//
// #include <stdio.h>
// #include <stdlib.h>
// #include <Xt/Intrinsic.h>
// #include <Xt/StringOps.h>
// #include <Xt/Widget.h>
// #include <Xt/Menu.h>
// #include <Xt/Popup.h>
// #include <Xt/Util.h>
//
// -----Global Variables-----
//
// Widget parent; /* The root widget */
// int start_up = 1; /* The flag to call init routine */
// GC line_gc; /* Pointer to GC */
// XColor *col; /* The GC values */
//
// -----Externs-----
//
// Routine: xt_redisplay(w, data, call_data)
// Description: Expose callback
// Return: Nothing
// Author: Rich Lee
//
// void xt_redisplay(w, data, call_data)
// Widget w; /* The widget associated with display */
// void *data; /* The data */
// XDrawnWidgetCallbackStruct *call_data; /* Not used */
// {
//     if (start_up)
//     {
//         start_up = 0;
//         xt_init();
//     }
//
//     main(argc, argv);
//     int argc;
//     char *argv[];
//     {
//         Arg args[10];

```

【図216】

```

//
// Filename: zero_vpp.C
// This routine contains the lines to find blocks that have non-zero vpp
// (level 1 segmentation)
//
// =====
//
#include "charlist.h"
#include "cimages.h"
//
// =====
// Scan a line quickly (every 1/4 pixel) looking for an 'on' pixel
int line_scan_quick( // Returns 1 for pixel found, 0 otherwise
    cimage& line,    // The line to process
    int c)           // The column to scan
{
    int r;
    for (r=0; r<line.sizeR; r++)
        if ((line.pixel[r][c] & 1)
            return 1;
    return 0;
}

// =====
// Scan a line completely looking for an 'on' pixel
int line_scan_complete( // Returns 1 for pixel found, 0 for otherwise
    cimage& line,        // The line
    int c)              // The column to scan
{
    int r;
    for (r=0; r<line.sizeR; r++)
        if ((line.pixel[r][c] & 1)
            return 1;
    return 0;
}

// =====
// The main routine. Call this to make the list
void extract_zero_vpp_list(
    cimage& line,        // The image of the line to segment
    struct NonZeroVppList *list) // A pointer to the list
{
    int c;
    int left, right;
    for (c=0; c<line.sizeC; c++) // Scan the image
    {
        if (line_scan_quick(line, c) // If found something
        {
            left = right = c;        // Look to the left for the start
            while (--c >= 0)
                if (line_scan_complete(line, c) == 0)
                    break;
            left = c+1;
            c = right;                // Keep looking to the right
            while (++c < line.sizeC)
                if (line_scan_complete(line, c) == 0)
                    break;
            right = c;
            zero_vpp_add(list, left, right); // Add in the new break
        }
    }
}

```

【図218】

```

CC=      cc
CFLAGS=  CC
COMPILE_PROFILE = -p
COMPILE_STYLE = -g -O
COMPILE_VISIBLE = -DVISIBLE
COMPILE_WARE_DBASE = -DWARE_DATABASE
LDLIBS= -lm
LDLIBS= -lXm -lXt -lX11 -l -lalloc -lPW
CPPFLAGS= -DCPP $(COMPILE_STYLE) $(COMPILE_VISIBLE) $(COMPILE_PROFILE) $(COMPILE_WARE_DBASE)
CFLAGS= $(COMPILE_STYLE) $(COMPILE_VISIBLE) $(COMPILE_PROFILE) $(COMPILE_WARE_DBASE)
OBJECTS_courcut.o psegment.o lineprocess.o xerovvp.o charlist.o cimages.o mymalloc.o
xs.o xt.o util.o autoinput.o courcut.o thisu_file.o

all: courcut

courcut: $(OBJECTS_courcut)
$(CXX) -o courcut $(OBJECTS_courcut) $(LDLIBS) $(LDLIBS)

autoinput.o: autoinput.c
$(CC) -c $(CFLAGS) autoinput.c

cimages.o: cimages.c
$(CXX) -c $(CXXFLAGS) cimages.c

courcut.o: courcut.C
$(CXX) -c $(CXXFLAGS) courcut.C

lineprocess.o: lineprocess.C
$(CXX) -c $(CXXFLAGS) lineprocess.C

mymalloc.o: mymalloc.c
$(CC) -c $(CFLAGS) mymalloc.c

psegment.o: psegment.C
$(CXX) -c $(CXXFLAGS) psegment.C

scaleprint.o: scaleprint.C
$(CXX) -c $(CXXFLAGS) scaleprint.C

thisu_file.o: thisu_file.c
$(CC) -c $(CFLAGS) thisu_file.c

util.o: util.c
$(CC) -c $(CFLAGS) util.c

xs.o: xs.c
$(CC) -c $(CFLAGS) xs.c

xt.o: xt.c
$(CC) -c -DCPP $(CFLAGS) xt.c

xerovvp.o: xerovvp.C
$(CXX) -c $(CXXFLAGS) xerovvp.C

charlist.o: charlist.h mymalloc.h
combine.o: charlist.h cimages.h
cimages.o: cimages.h xs.h
level4.o: mymalloc.h
lineprocess.o: charlist.h cimages.h
multilevel.o: charlist.h xs.h mymalloc.h
multilevel.o: charlist.h cimages.h xs.h mymalloc.h
outline.o: charlist.h cimages.h
psegment.o: cimages.h xs.h mymalloc.h
level3al.o: charlist.h cimages.h
xerovvp.o: charlist.h cimages.h

```

【図219】

```

/* filename: autoinput.h
 * This file contains the code to input from a filename
 * .....
#include <stdio.h>
#include "autoinput.h"

/* Global variables */
int auto_eof_detected;
FILE *auto_fp;
int save_to_file;
int save_count;
char filename[100];

.....
void auto_initialize()
{
    int not_exit = 1;

    auto_eof_detected = 0;

    while(not_exit)
    {
        printf("Input filename for auto-input? ");
        scanf("%s", filename);

        auto_fp = fopen(filename, "r");

        if (auto_fp == NULL) /* Read open failed */
        {
            auto_fp = fopen(filename, "w");
            if (auto_fp == NULL) /* Write open failed */
            {
                printf("Cannot open file to 'w', filename).\n");
                printf("As opened as a new file.\n", filename);
                printf("Enter 1 at the next prompt.\n");
                save_to_file = 1;
                auto_eof_detected = 1;
                not_exit = 0;
            }
            else /* Read open succeeded */
            {
                printf("Opening existing file %s\n", filename);
                not_exit = 0;
            }
        }

        printf("Do you wish to save the auto-input into this file? (1 = yes) ? ");
        save_to_file = auto_input_int();
        printf("Heard. Change the int '1' to '0' to turn off appending to the file.\n");
        save_count = 0;
    }
}

/* auto_input_int()
 * int i;
 * if (auto_eof_detected)
 * {
 *     printf("\nManual input (-9 = newline, -9999 = exit) ) ");

```

```

scanf("%d", &i);
if (i == -9999)
{
    fclose(auto_fp);
    exit(0);
}
if (save_to_file == 1)
{
    if (i == -9) /* Print new lines */
    {
        printf(auto_fp, "\n"); /* Print the new lines */
        fflush(auto_fp);
        save_count = 0;
    }
    return auto_input_int(); /* AND input again */
}
printf(auto_fp, "%d ", i);
fflush(auto_fp);
if (++save_count == 30)
{
    save_count = 0;
    printf(auto_fp, "\n");
}
return i;
}

if (fclose(auto_fp, "%d", &i) == EOF)
{
    printf("\n*** DO OF FILE DETECTED ***\n");
    if (save_to_file == 1)
    {
        fclose(auto_fp);
        auto_fp = fopen(filename, "a");
        if (auto_fp == NULL)
        {
            printf("***** WARNING: CANNOT APPEND TO FILE: %s ***\n", filename);
            save_to_file = 0;
        }
        printf(auto_fp, "\n");
    }
    auto_eof_detected = 1;
    return auto_input_int();
}
printf("%d\n", i);
return i;
}

void auto_terminate()
{
    fclose(auto_fp);
}

```

【図221】

```

/* Filename: Charlist.c
 * This file contains Charlist and Vector/ypoint utility routines
 * .....
 */
/* .....
 * Routines to handle Character list
 * .....
 */
/* By Christopher Sherrick */
#include "charlist.h"
#include "ymalloc.h"

/* .....
 * Initialize and allocate a Characterlist (set up curve pointers) */
void Charlist_initialize(char **clist, int clist_size, int clist_first, int clist_prev, int vector_height)
{
    int i;
    /* A looping var */

    /* Allocate memory for characters */
    clist = (char **) malloc((unsigned) clist_size * sizeof(struct Character));
    if (clist == NULL) {
        printf("Memory Allocation Error in Charlist_initialize()\n");
        exit(1);
    }
    clist_size = 0;
    clist_allocated = clist_size; /* Size Allocated */
    clist_first = 0; /* No first character */
    clist_vector_height = vector_height;

    /* Allocate memory for curves and set pointers */
    /* No memory to initialize */
    for (i=0; i<clist_size; i++) {
        clist[i].curve = i * vector_height;
        clist[i].curve = i * vector_height;
    }
}

/* .....
 * Zero a previously allocated Characterlist */
void Charlist_clear(struct Characterlist *clist)
{
    clist_size = 0;
    clist_first = 0;
    clist_prev = 0;
}

/* .....
 * Terminate & deallocate the Characterlist */
void Charlist_terminate(struct Characterlist *clist)
{
    int i;
    for (i=clist_allocated-1; i>=0; i--) { /* Free curves */
        free_vector(clist->curve[i].curve);
        free_vector(clist->curve[i].curve);
    }
}

```

```

}
free(struct Character *clist->next);
clist_size = 0;
clist_allocated = 0;
clist_first = 0;
}

/* .....
 * Place a new character in the charlist */
/* Takes the character beyond the charlist (charlist),
 * calculates loc, and inserts it into the list */
void Charlist_insert(struct Characterlist *clist, struct Character *newchar)
{
    struct Character *newchar; /* Pointer to the new character adding */
    newchar = (struct Character *) malloc(sizeof(struct Character)); /* Set newchar */
    /* Calculate the position of the new character */
    Charlist_recalculate_loc(newchar);

    /* Check for full list */
    if (clist_size == clist_allocated) /* If too big.. */
        printf("Error in Charlist_insert(): Character list is FULL!\n");
        exit(1);

    /* Place new in the list character */
    Charlist_locate_character(clist, newchar);

    /* .....
     * This routine repositions a character in the charlist. It removes
     * the character, recalculates loc and inserts it in the list accordingly
     */
    void Charlist_relocate_character(struct Characterlist *clist, struct Character *ch)
    {
        struct Character *newchar;
        struct Character *ch;

        /* Recalculate the position of the character */
        Charlist_recalculate_loc(ch);

        if (clist_size < 2) /* If it's the only item in the list */
            return;

        /* Remove the character from the list */
        if (ch->prev == 0) /* If it's the first item on list */
            clist_first = clist->first->next;
        else {
            ch->prev->next = ch->next;
            if (ch->next != 0)
                ch->next->prev = ch->prev;
        }

        /* Place the character in the list */
        Charlist_locate_character(clist, ch);
    }

    /* .....
     * Insert a character after doing a cut */
}

```

[ 2 2 2 ]

```

/* This routine will insert the character beyond the charlist (chr[size]).
 * loc is recalculated for both character and the new character, however
 * loc is NOT USED TO PLACE THE CHARACTER. It will
 * place the new character after character after finished, it is an
 * extremely good idea to reset the breakflag!
 */
void Charlist_Insert_After_Put(Charlist, ch, after)
{
    struct Charlist *charlist;
    struct Character *ch_after; /* Place the new character after this one */
    struct Character *newchar; /* Pointer to the new character adding */
    newchar = (struct Character *) malloc(sizeof(struct Character)); /* Set newchar */

    /* Calculate the position of the new character */
    Charlist_Reset_List_Location(charlist, loc);
    Charlist_Reset_List_Location(charlist, loc);

    /* Check for full list */
    if (loc == size) /* Increment the size of the list */
    {
        loc = size + 1;
        if (loc == size + 1) /* If too big... */
        {
            printf("Error in Charlist_Place_New_Char(): Character list is FULL!\n");
            exit(1);
        }
    }

    newchar->prev = ch_after;
    newchar->next = ch_after->next;
    ch_after->next = newchar;
    if (newchar->next != 0)
        newchar->next->prev = newchar;
}

/* This routine will remove a character from the charlist
 * Note: The memory is NOT reclaimed! so don't rely on this too much.
 * Also, prev and next pointers of ch are NOT altered!
 */
void Charlist_Remove(Charlist, ch)
{
    struct Charlist *charlist;
    struct Character *ch;

    /* Remove the character from the list */
    if (ch == 0) /* If list on list */
    {
        charlist->first = charlist->first->next;
        charlist->first->prev = 0;
    }
    else
    {
        ch->prev->next = ch->next;
        if (ch->next != 0)
            ch->next->prev = ch->prev;
    }
}

/* This routine will recalculate the position of a character */
void Charlist_Reset_List_Location(struct Charlist *charlist, int position)
{
    struct Character *ch; /* The character */

    int i, loc;
    /* Calculate the position of the new character */
    loc = 0;
    for (i = 0; i < charlist->size; i++) /* Position - average position.. */
        loc += charlist->list[i].ch - charlist->list[i].ch; /* .. of the curves */
    loc = loc * 4 / (charlist->size - charlist->size); /* Set it */
    charlist->loc = loc;
}

```

```

/* This routine will place a character (that is not in the charlist)
 * into the charlist.
 */
void Charlist_Place_Character(Charlist, ch)
{
    struct Charlist *charlist; /* The character */
    struct Character *p; /* Used for searching for location */
    struct Character *last_p; /* The character before p */
    int loc;

    /* Is the list empty? */
    if (loc == 0)
    {
        ch->next = 0;
        ch->prev = 0;
        charlist->first = ch;
        return;
    }

    /* Possible optimization: don't start from start of list...
     * start from where at */
    loc = ch->loc; /* Retrieve position */

    /* Is it at the beginning of the list? */
    p = charlist->first; /* Load p with the first item on the list */
    if (loc < p->loc) /* If it's the first on the list */
    {
        p->prev = ch;
        charlist->first = ch; /* Set the pointers */
        ch->next = p;
        ch->prev = 0;
        return;
    }

    /* Set the list on the list to this */
    while (p->loc <= loc) /* Until find record dist where to place */
    {
        last_p = p; /* Remember last position */
        p = p->next; /* Get the next character */
        if (p == 0) /* If it's the end of list, insert it there */
            break;
    }

    ch->prev = last_p; /* Insert new position after last_p */
    last_p->next = ch;
    if (p != 0) /* If nextchar isn't at the end of the list */
        p->prev = ch;
}

/* Allocate memory for list */
size_t size; /* Compute memory allocated, etc */
charlist = (struct Character *) malloc(size); /* Allocate memory */
charlist->list = (struct Character *) malloc(size); /* Set max allocated */
charlist->size = 0; /* Set to zero elements */

```



1

—403—

【図225】

```

// pixel = (unsigned char *)malloc((unsigned int) size*sizeof(unsigned char)); //
// Allocate memory
if (pixel == NULL) // Allocate memory
{
    cerr << "Error: Memory Allocation Error (out of memory)\n";
    exit(1);
}
for (int i = 0; i < size, i++)
{
    pixel[i] = (unsigned char *)malloc((unsigned int) size - sizeof(unsigned char));
    // Allocate core mem
    if (pixel[i] == NULL)
    {
        cerr << "Error: Memory Allocation Error (out of memory)\n";
        exit(1);
    }
}
allocated_size = size;
allocated_sizeC = sizeC;
}

// Deallocate memory for an image from memory
void cimage::deallocate()
{
    if ((allocated_size != 0) && (allocated_sizeC != 0))
    {
        for (int i = 0; i < allocated_size; i++)
        {
            free(pixel[i]);
            if (sizeC != 0) // Zero out the size
            {
                size = 0;
                sizeC = 0;
                allocated_size = 0;
                allocated_sizeC = 0;
            }
        }
    }
}

// This allow allocation of memory from outside the class. It will
// Allocate memory if it's larger. If not, it will just adjust the size
void cimage::reallocate(int R, int C)
{
    int new_R, new_C;
    if ((R > allocated_size) || (C > allocated_sizeC))
    {
        new_R = allocated_size;
        new_C = allocated_sizeC;
        if (R > new_R)
            new_R = R;
        if (C > new_C)
            new_C = C;
        allocate(new_R, new_C); // Re-allocate the image
        sizeR = R;
        sizeC = C;
    }
}

// Create a class with image value, value, x and y are the size
// void cimage::cimage(char value, int R, int C)

```

```

// Filename: cimage.cpp
// This file contains a vast array image utility functions, for a cimage
//
//
//
// Undelete this to make images.c run invisibly (except for error)
// Define PRINT_IMAGES_INFO
#include <image.h>
#include <stdio.h>
//... CLASS: cimage (in array of char)
//.....
cimage::cimage()
{
    sizeR = 0;
    sizeC = 0;
    allocated_sizeR = 0;
    allocated_sizeC = 0;
    dpi = 0;
}

cimage::cimage(int r, int c)
{
    sizeR = 0;
    sizeC = 0;
    allocated_sizeR = 0;
    allocated_sizeC = 0;
    dpi = 0;
    allocate(r, c);
}

cimage::cimage(cimage img)
{
    allocate(sizeR, sizeC); // Allocate memory for new cimage (loads sizes)
    for (int i = 0; i < sizeR; i++)
    {
        for (int j = 0; j < sizeC; j++) // Copy old image to new cimage
            pixel[i][j] = img.pixel[i][j];
    }
}

// Delete the cimage
cimage::~cimage()
{
    deallocate();
}

// Allocate memory for an image from memory
void cimage::allocate(int r, int c)
{
    deallocate(); // Remove current cimage
    if (r < 0 || c < 0)
    {
        cerr << "Illegal cimage size: " << r << " by " << c << "\n";
        exit(1);
    }
    sizeR = r;
    sizeC = c;
    // Set cimage sizes
    if ((r != 0) && (c != 0))

```

【図226】

```

        for (j=C1; j<C2; j++)
            temp_pixel[(i-1)*C1+j-C1] = pixel[i][j];
        allocate(temp_size, temp_sizeC); // Copy temp image to current image
        for (l=0; l<sizeC; l++)
            temp_pixel[l] = pixel[i][l];
        pixel[i][j] = temp_pixel[i][j];
    }

    // Read in the TIFF file
    // Read in the TIFF file
    void change_readoff(char *filename;
    {
        FILE *tagfile; // Input file
        long int tagfile_position; // The our file position (in bytes)
        long int width; // Width of picture
        long int height; // Height of picture
        long int bps; // Value of black pixel
        long int bps; // Bytes per strip
        long int raster_offset; // Start of image data
        long int numtags, tag, bytes, len, value; // Tag values
        long int i, j;

        if ((tagfile = fopen(filename, "rb")) == NULL)
        {
            cout << "An error in opening a file name << 'n';
            exit(1);
        }
        tagfile_position = 0;

        // Read in header and make sure it's Intel
        i = getbytes(tagfile, tagfile_position);
        if (i != 0x002A)
        {
            cout << "... Error: TIFF file not in Intel format\n";
            exit(1);
        }

        // Read in magic number
        i = getbytes(tagfile, tagfile_position);
        if (i != 0x002A)
        {
            cout << "BAD Magic Number for TIFF file!\n";
            exit(1);
        }

        // Find first file directory
        j = getbytes(tagfile, tagfile_position);
        while(tagfile_position < j)
        {
            i = getbytes(tagfile, tagfile_position);
            // Tag in directory
            numtags = getbytes(tagfile, tagfile_position);
            while (i-->numtags > 0) {
                tag = getbytes(tagfile, tagfile_position);
                type = getbytes(tagfile, tagfile_position);
                len = getbytes(tagfile, tagfile_position);
                value = getbytes(tagfile, tagfile_position);
                if ((len % 4) != 0) {tag & 32768} == 0)
                {
                    cout << "Error: Tag length is 1\n";
                    exit(1);
                }
            }
        }
    }
}

```

—406—

```

while(tagfile_position < raster_offset)
{
    i = getbytes(tagfile, tagfile_position);
}

//----- READ IN FILE
allocate(int length, int width);

int v;

for (i = 0, i=length, i++)
{
    for (j = 0, j=width, j++)
    {
        if (i%30 == 0)
        {
            v = getc(tagfile);
            tagfile_position++;
            if (v == EOF)
            {
                cerr << "\n Error: Premature End of File found in tiff file\n";
                exit(1);
            }
            if ((v & 128) == black(122))
            {
                black[i][j] = 0;
            }
            else
            {
                pixel[i][j] = 1;
            }
        }
    }
}

fclose(tagfile);

// Read in a byte, file position is updated
int ci = 0;
int ci2 = 0;
int ci3 = 0;
int ci4 = 0;
int ci5 = 0;
int ci6 = 0;
int ci7 = 0;
int ci8 = 0;
int ci9 = 0;
int ci10 = 0;
int ci11 = 0;
int ci12 = 0;
int ci13 = 0;
int ci14 = 0;
int ci15 = 0;
int ci16 = 0;
int ci17 = 0;
int ci18 = 0;
int ci19 = 0;
int ci20 = 0;
int ci21 = 0;
int ci22 = 0;
int ci23 = 0;
int ci24 = 0;
int ci25 = 0;
int ci26 = 0;
int ci27 = 0;
int ci28 = 0;
int ci29 = 0;
int ci30 = 0;
int ci31 = 0;
int ci32 = 0;
int ci33 = 0;
int ci34 = 0;
int ci35 = 0;
int ci36 = 0;
int ci37 = 0;
int ci38 = 0;
int ci39 = 0;
int ci40 = 0;
int ci41 = 0;
int ci42 = 0;
int ci43 = 0;
int ci44 = 0;
int ci45 = 0;
int ci46 = 0;
int ci47 = 0;
int ci48 = 0;
int ci49 = 0;
int ci50 = 0;
int ci51 = 0;
int ci52 = 0;
int ci53 = 0;
int ci54 = 0;
int ci55 = 0;
int ci56 = 0;
int ci57 = 0;
int ci58 = 0;
int ci59 = 0;
int ci60 = 0;
int ci61 = 0;
int ci62 = 0;
int ci63 = 0;
int ci64 = 0;
int ci65 = 0;
int ci66 = 0;
int ci67 = 0;
int ci68 = 0;
int ci69 = 0;
int ci70 = 0;
int ci71 = 0;
int ci72 = 0;
int ci73 = 0;
int ci74 = 0;
int ci75 = 0;
int ci76 = 0;
int ci77 = 0;
int ci78 = 0;
int ci79 = 0;
int ci80 = 0;
int ci81 = 0;
int ci82 = 0;
int ci83 = 0;
int ci84 = 0;
int ci85 = 0;
int ci86 = 0;
int ci87 = 0;
int ci88 = 0;
int ci89 = 0;
int ci90 = 0;
int ci91 = 0;
int ci92 = 0;
int ci93 = 0;
int ci94 = 0;
int ci95 = 0;
int ci96 = 0;
int ci97 = 0;
int ci98 = 0;
int ci99 = 0;
int ci100 = 0;
int ci101 = 0;
int ci102 = 0;
int ci103 = 0;
int ci104 = 0;
int ci105 = 0;
int ci106 = 0;
int ci107 = 0;
int ci108 = 0;
int ci109 = 0;
int ci110 = 0;
int ci111 = 0;
int ci112 = 0;
int ci113 = 0;
int ci114 = 0;
int ci115 = 0;
int ci116 = 0;
int ci117 = 0;
int ci118 = 0;
int ci119 = 0;
int ci120 = 0;
int ci121 = 0;
int ci122 = 0;
int ci123 = 0;
int ci124 = 0;
int ci125 = 0;
int ci126 = 0;
int ci127 = 0;
int ci128 = 0;
int ci129 = 0;
int ci130 = 0;
int ci131 = 0;
int ci132 = 0;
int ci133 = 0;
int ci134 = 0;
int ci135 = 0;
int ci136 = 0;
int ci137 = 0;
int ci138 = 0;
int ci139 = 0;
int ci140 = 0;
int ci141 = 0;
int ci142 = 0;
int ci143 = 0;
int ci144 = 0;
int ci145 = 0;
int ci146 = 0;
int ci147 = 0;
int ci148 = 0;
int ci149 = 0;
int ci150 = 0;
int ci151 = 0;
int ci152 = 0;
int ci153 = 0;
int ci154 = 0;
int ci155 = 0;
int ci156 = 0;
int ci157 = 0;
int ci158 = 0;
int ci159 = 0;
int ci160 = 0;
int ci161 = 0;
int ci162 = 0;
int ci163 = 0;
int ci164 = 0;
int ci165 = 0;
int ci166 = 0;
int ci167 = 0;
int ci168 = 0;
int ci169 = 0;
int ci170 = 0;
int ci171 = 0;
int ci172 = 0;
int ci173 = 0;
int ci174 = 0;
int ci175 = 0;
int ci176 = 0;
int ci177 = 0;
int ci178 = 0;
int ci179 = 0;
int ci180 = 0;
int ci181 = 0;
int ci182 = 0;
int ci183 = 0;
int ci184 = 0;
int ci185 = 0;
int ci186 = 0;
int ci187 = 0;
int ci188 = 0;
int ci189 = 0;
int ci190 = 0;
int ci191 = 0;
int ci192 = 0;
int ci193 = 0;
int ci194 = 0;
int ci195 = 0;
int ci196 = 0;
int ci197 = 0;
int ci198 = 0;
int ci199 = 0;
int ci200 = 0;
int ci201 = 0;
int ci202 = 0;
int ci203 = 0;
int ci204 = 0;
int ci205 = 0;
int ci206 = 0;
int ci207 = 0;
int ci208 = 0;
int ci209 = 0;
int ci210 = 0;
int ci211 = 0;
int ci212 = 0;
int ci213 = 0;
int ci214 = 0;
int ci215 = 0;
int ci216 = 0;
int ci217 = 0;
int ci218 = 0;
int ci219 = 0;
int ci220 = 0;
int ci221 = 0;
int ci222 = 0;
int ci223 = 0;
int ci224 = 0;
int ci225 = 0;
int ci226 = 0;
int ci227 = 0;
int ci228 = 0;
int ci229 = 0;
int ci230 = 0;
int ci231 = 0;
int ci232 = 0;
int ci233 = 0;
int ci234 = 0;
int ci235 = 0;
int ci236 = 0;
int ci237 = 0;
int ci238 = 0;
int ci239 = 0;
int ci240 = 0;
int ci241 = 0;
int ci242 = 0;
int ci243 = 0;
int ci244 = 0;
int ci245 = 0;
int ci246 = 0;
int ci247 = 0;
int ci248 = 0;
int ci249 = 0;
int ci250 = 0;
int ci251 = 0;
int ci252 = 0;
int ci253 = 0;
int ci254 = 0;
int ci255 = 0;
int ci256 = 0;
int ci257 = 0;
int ci258 = 0;
int ci259 = 0;
int ci260 = 0;
int ci261 = 0;
int ci262 = 0;
int ci263 = 0;
int ci264 = 0;
int ci265 = 0;
int ci266 = 0;
int ci267 = 0;
int ci268 = 0;
int ci269 = 0;
int ci270 = 0;
int ci271 = 0;
int ci272 = 0;
int ci273 = 0;
int ci274 = 0;
int ci275 = 0;
int ci276 = 0;
int ci277 = 0;
int ci278 = 0;
int ci279 = 0;
int ci280 = 0;
int ci281 = 0;
int ci282 = 0;
int ci283 = 0;
int ci284 = 0;
int ci285 = 0;
int ci286 = 0;
int ci287 = 0;
int ci288 = 0;
int ci289 = 0;
int ci290 = 0;
int ci291 = 0;
int ci292 = 0;
int ci293 = 0;
int ci294 = 0;
int ci295 = 0;
int ci296 = 0;
int ci297 = 0;
int ci298 = 0;
int ci299 = 0;
int ci300 = 0;
int ci301 = 0;
int ci302 = 0;
int ci303 = 0;
int ci304 = 0;
int ci305 = 0;
int ci306 = 0;
int ci307 = 0;
int ci308 = 0;
int ci309 = 0;
int ci310 = 0;
int ci311 = 0;
int ci312 = 0;
int ci313 = 0;
int ci314 = 0;
int ci315 = 0;
int ci316 = 0;
int ci317 = 0;
int ci318 = 0;
int ci319 = 0;
int ci320 = 0;
int ci321 = 0;
int ci322 = 0;
int ci323 = 0;
int ci324 = 0;
int ci325 = 0;
int ci326 = 0;
int ci327 = 0;
int ci328 = 0;
int ci329 = 0;
int ci330 = 0;
int ci331 = 0;
int ci332 = 0;
int ci333 = 0;
int ci334 = 0;
int ci335 = 0;
int ci336 = 0;
int ci337 = 0;
int ci338 = 0;
int ci339 = 0;
int ci340 = 0;
int ci341 = 0;
int ci342 = 0;
int ci343 = 0;
int ci344 = 0;
int ci345 = 0;
int ci346 = 0;
int ci347 = 0;
int ci348 = 0;
int ci349 = 0;
int ci350 = 0;
int ci351 = 0;
int ci352 = 0;
int ci353 = 0;
int ci354 = 0;
int ci355 = 0;
int ci356 = 0;
int ci357 = 0;
int ci358 = 0;
int ci359 = 0;
int ci360 = 0;
int ci361 = 0;
int ci362 = 0;
int ci363 = 0;
int ci364 = 0;
int ci365 = 0;
int ci366 = 0;
int ci367 = 0;
int ci368 = 0;
int ci369 = 0;
int ci370 = 0;
int ci371 = 0;
int ci372 = 0;
int ci373 = 0;
int ci374 = 0;
int ci375 = 0;
int ci376 = 0;
int ci377 = 0;
int ci378 = 0;
int ci379 = 0;
int ci380 = 0;
int ci381 = 0;
int ci382 = 0;
int ci383 = 0;
int ci384 = 0;
int ci385 = 0;
int ci386 = 0;
int ci387 = 0;
int ci388 = 0;
int ci389 = 0;
int ci390 = 0;
int ci391 = 0;
int ci392 = 0;
int ci393 = 0;
int ci394 = 0;
int ci395 = 0;
int ci396 = 0;
int ci397 = 0
```

【図229】

```

// Filename: cimages.h
// This is the header file for cimages.C
//
//
// =====
//
// #define NOX_DEBUG 1
// #include <stream.h>
// #include <stdio.h>
// #include <stdlib.h>
// #include <math.h>

class cimage; // Forward reference

// =====
// *** CLASS: cimage
// =====
class cimage
{
public:
    int      sizeR; // Number of rows (first index of array)
    int      sizeC; // Number of cols. (second index of array)
    int      allocated_sizeR; // Amount allocated
    int      allocated_sizeC; // Amount allocated
    int      dpi; // Dots per inch of the image (0 if unknown)
    unsigned char *pixel; // The binary image (2d array)

    cimage(); // Initialise
    ~cimage(); // Destroy
    cimage(int r, int c); // Initialise with a size
    cimage(cimage& temp);

    void reallocate(int R, int C); // Will allocate memory for an image
    void fill(char value, int R, int C); // Fill image with a constant value
    void clip(cimage& image, int R1, int R2, int C1, int C2);
    void clipmask(); // Clips zero space surrounding image
    void readtiff(char *filename); // Read in a tiff image
    void plot(int x, int y); // Plot binary image
    void plotbox(int x, int y); // Plot with a box around it

private:
    void allocate(int r, int c); // Allocate memory
    void deallocate(); // Deallocate memory
    int getbyte(FILE *f, long int& file_position, int eoferror);
    long int get2bytes(FILE *f, long int& fp);
    long int get4bytes(FILE *f, long int& fp);
};

```

—408—

```

// Filename: countcut.c
// This routine cuts coursetr chars (output from level 1)
// (level 1 segmentation)
//
// Includes: countcut.h
// Defines globals:
//
// int COUNT_CUT_WIDTH;
// int COUNT_CUT_HEIGHT;
//
// This routine will take care of the non-zero vpp list
// void count_cut_process()
// {
//     struct histogram *hist; // The line
//     struct histogram *old_hist; // Range to process
//     int start_col;
//     int last_col;
//     int last_row;
//     int cut_value;
//     int cut_at;
//     int hist[10];
//     int old_hist[10];
//     int num_lines;
//     int num_cols;
//     int z_c;
//
//     // Calculate num_cols = width - step_col - start_col; // The width
//     if (width <= COUNT_CUT_AVC_WIDTH + COUNT_CUT_DELTA)
//         num_cols = 1;
//     else // It's only one character
//         num_cols = (int)((float)width / ((float)COUNT_CUT_AVC_WIDTH + 0.1));
//
//     printf("COUNT CUT BLOCK WIDTH= %d => # of characters= %n", width, num_cols);
//
//     for (cut_col = start_col; cut_col < start_col + num_cols; cut_col++)
//     {
//         // Break up the region
//         cr_start = cut_col;
//         cr_stop = cut_col + COUNT_CUT_HEIGHT;
//         cr_step = cut_col - COUNT_CUT_DELTA;
//         if (cr_start < start_col) // Make sure it doesn't go off edges
//             cr_start = start_col;
//         if (cr_stop > stop_col)
//             cr_stop = stop_col;
//
//         // Make histogram over this range
//         for (c=cr_start; c<cr_stop; c++)
//         {
//             hist[c-cr_start] = 0;
//             for (l=0; l<line_size; l++)
//                 if (line_pixel[l][c] & 1)
//                     hist[c-cr_start]++;
//         }
//
//         Find minimum in histogram
//         min_val = 9999;
//         min_col = cr_start;
//         for (l=0; l<line_size; l++)
//         {
//             if (line_pixel[l][min_col] < min_val)
//                 min_val = line_pixel[l][min_col];
//             min_col++;
//         }
//
//         // This routine will take each place from the input image and send it
//         // to be processed by the courier cut routine
//         void courier_cut()
//         {
//             struct histogram *hist; // The image of the line to segment
//             struct histogram *old_hist; // A pointer to the old list (input)
//             struct histogram *new_hist; // A pointer to the new list (output)
//
//             int i;
//
//             hist = malloc(sizeof(hist)); // Allocate the list;
//             for (i=0; i<old_hist->nlines; i++)
//                 coutcur_hist[i], old_hist->list[i+1]);
//         }
//
//         // ***** WITH *****
//         // To be processed by the courier cut routine
//         void courier_cut()
//         {
//             struct histogram *hist; // The image of the line to segment
//             struct histogram *old_hist; // A pointer to the old list (input)
//             struct histogram *new_hist; // A pointer to the new list (output)
//
//             int i;
//
//             hist = malloc(sizeof(hist)); // Allocate the list;
//             for (i=0; i<old_hist->nlines; i++)
//                 coutcur_hist[i], old_hist->list[i+1]);
//         }
//
//         // Add the cut and continue
//         zerovpp_addinlist, last_cout, cut_at;
//         last_cout = cut_at;
//         // Insert last cut
//         zerovpp_addinlist, last_cout, stop_col;
//
//         printf(" Cut at %ld, (searched to +/- %ld + (%ld*%ld)/n",
//             cut_at-start_col, cut_pos-start_col, COUNT_CUT_DELTA,
//             cr_start-start_col, cr_stop-start_col);
//         cut_at = 1;
//     }
// }

```

[illegible]

[illegible]



【図233】

```

/* This program contains common vector and matrix macros */
#include <stdio.h>
#include <stdlib.h>
/* Routine for error during memory allocation */
void allocation_error()
{
    char *s;
    /* The error text */
    s = "coredump";
    printf("Memory allocation error %s\nProgram Aborted.\n", s);
    coredump = 0;
    coredump = 0;
}

/* Allocate an integer vector of size [0..n-1] */
int *ivector(n)
{
    int i;
    /* Size of vector */
    i = n;
    v = (int *)malloc((unsigned) n*sizeof(int));
    if (!v) allocation_error("in ivector()");
    return v;
}

/* Free an integer vector v of size [0..n-1] */
void free_ivector(v)
{
    int *v;
    /* The vector */
    free((int *) v);
}

/* Allocate an float vector of size [0..n-1] */
float *fvector(n)
{
    float *v;
    /* Size of vector */
    int n;
    v = (float *)malloc((unsigned) n*sizeof(float));
    if (!v) allocation_error("in fvector()");
    return v;
}

/* Free an float vector v of size [0..n-1] */
void free_fvector(v)
{
    float *v;
    /* The vector */
    free((float *) v);
}

/* Allocate a ranged vector v of size [m..n-1] */
int *ivector_ranged(m,n)
{
    int i;
    /* The range */
    int *v;
    v = (int *)malloc((unsigned) (n-m)*sizeof(int));
    if (!v) allocation_error("in ivector_ranged()");
    return (v+m);
}

/* Free an integer ranged vector v of size [m..n-1] */
void free_ivector_ranged(v,m)
{
    int *v;
    /* The vector */
    int m;
    /* The starting index of v */
    free((int *) (v-m));
}

/* Allocate an integer matrix of size [0..r-1][0..c-1] */
int **matrix(r,c)
{
    int r,c;
    /* The size of the matrix */
    int **m;
    int i;
    m = (int **)malloc((unsigned) r * sizeof(int *));
    if (!m) allocation_error("in matrix()");
    for (i=0; i<r; i++) {
        m[i] = (int *)malloc((unsigned) c * sizeof(int));
        if (!m[i]) allocation_error("in matrix()");
    }
    return m;
}

/* Free an integer matrix m of size [0..r-1][0..c-1] */
void free_matrix(m,r)
{
    int **m;
    /* The matrix */
    int r;
    /* The size (num rows) */
    for (i=0; i<r; i++) {
        free((int *) m[i]);
    }
    free((int **) m);
}

/* Allocate an char matrix of size [0..c-1][0..r-1] */
char **cmatrix(r,c)
{
    int r,c;
    /* The size of the matrix */
    char **m;
    int i;
    m = (char **)malloc((unsigned) r * sizeof(char *));
    if (!m) allocation_error("in cmatrix()");
    for (i=0; i<r; i++) {
        m[i] = (char *)malloc((unsigned) c * sizeof(char));
        if (!m[i]) allocation_error("in cmatrix()");
    }
    return m;
}

/* Free an character matrix m of size [0..r-1][0..c-1] */
void free_cmatrix(m,r)
{
    char **m;
    /* The matrix */
    int r;
    /* The size (num rows) */
    for (i=0; i<r; i++) {
        free((char *) m[i]);
    }
}

```

```

// Filenames permanent.c
// This routine contains stuff to do page segmentation
// .....
// .....
// .....
#define MAX_CHARS_PER_LINE 255
#define MAX_COLUMNS_PER_PAGE 80

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
#include<math.h>
#include<limits.h>
#include<unistd.h>

void process_line(charpage_image);
void setup_plot_line(charpage_image, int start_row);
int setup_plot_page(charpage_image, int start_row);

define NUM_ROWS 15 // The number of columns to divide the page into (original 45)
define COLUMNS_OVERLAP 0.3 // What fraction of two lines must overlap (original = 0.3)
define PLOT_OVERFLOW_MARGIN_PIXELS 0.034 // Minimum line height (inches)
define BLD (30.0/3.14159) // Radius to squares
define OVERLAP 1 // Half the width of the overlap

// Define external functions
extern void set_font_size(int fsize); // The line being segmented (so not remapped)
extern void set_font_size(int fsize); // Load in the font

// Global variables for page size (eliminate border noise)
int PAGE_W;
int PAGE_H;
int PAGE_X;
int PAGE_Y;
int PAGE_Z;
int PAGE_C;
int PAGE_O;

// Define temporary global variables
int WINDOW_SIZE_X;
int WINDOW_SIZE_Y;
int cursor_x;
int cursor_y;
extern int CURSOR_WIDTH;
extern int CURSOR_HEIGHT;

// ===== This routine will plot a scaled image on the screen =====
void clear_screen_and_plot_image()
{
    float scale, scale2;
}

// Zero before PAGE_X
// Histogram between X1 and X2
// (x(x) - 0).

```

【図237】

```

int end_of_list,
int row_div, // Loop termination flag
int range1, range2, // The test range in the topmost text
int r1, r2, c1, c2, // The test range in the adjacent division (row 1-a)
int min_line_height, // The minimum line height
int max_line_height, // The maximum line height
int i, j, b, r, o, // Misc handy variable

stopcol = (vector<INDIVISIONS>);
stopcol.resize(INDIVISIONS);
startcol = (vector<INDIVISIONS>);
startcol.resize(INDIVISIONS);
stopcol = (vector<INDIVISIONS>);
stopcol.resize(INDIVISIONS);
size = (vector<INDIVISIONS>);
size.resize(INDIVISIONS);
include = (vector<INDIVISIONS>);
include.resize(INDIVISIONS);

min_line_height = (int) ((float) page_cpi * PAGE_DP2_MIN_BLOCK_SIZE);

// Step 1: Define columns and find histograms
startcol[0] = PAGE_C1;
for (i=0; i<INDIVISIONS-1; i++)
{
    c = PAGE_C1 + (PAGE_C2 - (i+1)) / INDIVISIONS;
    startcol[i+1] = c - OVERLAP;
    stopcol[i] = c + OVERLAP;
}
stopcol[INDIVISIONS-1] = PAGE_C2;
// Find histograms
for (i=0; i<INDIVISIONS; i++)
{
    make_histogram_page, startcol[i], stopcol[i], ngram[i];
}

// Step 2: Make test area lists
for (i=0; i<INDIVISIONS; i++)
{
    // Note: Zeros are guaranteed on both ends of the histogram.
    // Initialize
    for (j=PAGE_R1-1; j<PAGE_R2; j++)
    {
        if (ngram[i][j] == 0) // If starting text
        {
            startcol[i][j] = j+1;
        }
        if (ngram[i][j] != 0) // If stopping text
        {
            stopcol[i][j] = j+1;
            size[i] = j;
        }
    }
}

// Step 3: Find the lines
for (i=0; i<INDIVISIONS; i++)
{
    cursor[i] = 0; // Set all pointers to the top of the list
    end_of_list = 0;
    while(end_of_list == 0)
    {
        // Step 3a: Find the highest text
        a = 1; // a = 1 page-align
        for (i=0; i<INDIVISIONS; i++)
        {
            if (cursor[i] < size[i]) // If valid cursor
            {
                b = startcol[i][cursor[i]] + stopcol[i][cursor[i]];
            }
        }
    }
}

```

```

for (c=c1; c<=c2; c++)
{
    if (page_align[c] != 0)
    {
        a[c] = 0; // Exit this search
        c = n;
    }
}
// Step 3b: Find the lines
for (i=0; i<INDIVISIONS; i++)
{
    cursor[i] = 0; // Set all pointers to the top of the list
    end_of_list = 0;
    while(end_of_list == 0)
    {
        // Step 3a: Find the highest text
        a = 1; // a = 1 page-align
        for (i=0; i<INDIVISIONS; i++)
        {
            if (cursor[i] < size[i]) // If valid cursor
            {
                b = startcol[i][cursor[i]] + stopcol[i][cursor[i]];
            }
        }
    }
}

```

【图 2 3 8】

-414-

【図239】

```

int border_ignore;
int i;

Charlist_initialize(&clist, MAX_CHARS_PER_LINE, MAX_LINE_HEIGHT);
Process_line.fill(0,40,350); // Allocate the memory for the lines
xs_win_dim(&WINDOW_SIZE_X, &WINDOW_SIZE_Y); // Get the window size

printf("Input tiff filename? /d2/tif/");
strcpy(filename, "/d2/tif/coshill.tif");
scanf("%s", filename);
strcpy(filename, "/d2/tif/");
strcat(filename, filename);

auto_initialize(); // Initialize Auto-Input routines

printf("Enter courtcut average width? ");
COURTCUT_AVC_WIDTH = auto_input_int();

printf("Enter courtcut delta? ");
COURTCUT_DELTA = auto_input_int();

page.dpi = 300;

border_ignore = 1;
// cout << "Input border width to ignore (in pixels)? ";
// cin >> border_ignore;

printf("Reading in File... ");
page.read(tiff(filename));
printf("Done.\n");

// sepp_init(page); // Initialize page printing routines
// border_ignore = (int) ((float) page.dpi * DPI_BORDER_IGNORE);
// printf("Border ignore = %d\n", border_ignore);

PAGE_R1 = border_ignore; // Define page size
PAGE_R2 = page.sizeR - border_ignore;
PAGE_AD = PAGE_R2 - PAGE_R1;
PAGE_C1 = border_ignore;
PAGE_C2 = page.sizeC - border_ignore;
PAGE_CD = PAGE_C2 - PAGE_C1;

// ***** THIS IS JUST FOR TESTING *****
xs_flush();
printf("Enter 0 to continue? ");
i = auto_input_int();

// clear_scale_and_plot_image(page);

printf("Pausing because page is displayed. Input 0 to segment page? ");
border_ignore = auto_input_int();

while(i) {
    segment_page(page, &clist);
    xs_flush();

    printf("Program finished. Enter 0 to run again? ");
    int ttemp;
    ttemp = auto_input_int();
}

Charlist_terminate(&clist);
auto_terminate();
}

```

-416-

```

./
./outline: ut_getbits.c.p.n)
./
./Description: Check if the bit is turned on or not
./
./Return: Nothing
./
./Author: Rick Lee
./
./ut_getbits.c.p.n)

```

【図242】

```

/*
File: xs.c
Comments:
This file contains routines for displaying 'things'.

xs_flush() y);
xs_flush() y);
xs_redisplay(x,y);
xs_redisplay();
xs_dis_bmap(x,y,width,height,buf);
xs_dis_wmap(x,y,width,height);
xs_dis_bmap(x,y,width,height);
xs_dis_wmap(x,y,width,height);
xs_dis_bmap(x,y,width,height);
xs_dis_wmap(x,y,width,height);

Usage:
Just call these routines once the root has been made with the
display routine.

*/
/* -----Includes----- */
#include <stdio.h>
#include <stdlib.h>
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Xatom.h>
#include <X11/Xproto.h>
#include <X11/Xext.h>
#include <X11/Xrandr.h>
#include <X11/Xrandr.h>
#include <X11/Xrandr.h>
/* -----Global Var----- */

/* -----Externs----- */
extern Widget parent; /* the root widget */
extern GC line_gc; /* the display widget */
extern XGCValues gc;

xs_flush()
{
/* Flush the queue to i-server */
XFlush(XDisplay(canvas));
}

/*
Routine: xs_display(x,y)
Description: Draw a point on the screen
Return: Nothing
Author: Rick Lee
*/
xs_display(x,y)
int x;
int y;
{
XDrawPoint(XDisplay(canvas), XWindow(canvas), line_gc, x, y);
}

/*
Routine: xs_unplot(x,y)
Description: Unplot a point on the screen
Return: Nothing
Author: Rick Lee
*/
xs_unplot(x,y)
int x;
int y;
{
XSetForeground(XDisplay(canvas), line_gc, gc.background);
XDrawPoint(XDisplay(canvas), XWindow(canvas), line_gc, x, y);
XSetForeground(XDisplay(canvas), line_gc, gc.foreground);
}

/*
Routine: xs_redisplay()
Description: Redisplay the 'stuff' that was on the window
Return: Nothing
Author: Rick Lee
*/
xs_redisplay()
{
}

/*
Routine: xs_dis_bmap(x,y,width,height,buf)
Description: Draw the given bitmap
Return: Nothing
Author: Rick Lee
*/
xs_dis_bmap(x,y,width,height,buf)
Pixmap bitmap, pix, /* The bitmap for display */
/* Reverse the bit ordering */
/* Reverse width,height,buf */
/* First create bitmap */
Bitmap = XCreateBitmapFromData(XDisplay(canvas), XScreen(canvas),
buf,width,height);
/* Create the pixmap area */
pix = XCreatePixmap(XDisplay(canvas), XWindow(canvas), width,height,
eight);
DefaultDepthOfScreen(XScreen(canvas));
/* Now copy the data to bitmap */
XCopyArea(XDisplay(canvas), bitmap, pix, line_gc, 0,
width,height,0,0);
/* Now copy the pixmap to drawing area */
}

```

【図243】

```

XCopyArea(XtDisplay(canvas), pix, XtWindow(canvas), line_gc, 0, 0,
          width, height, x, y);

/* Now free the bitmap */
XFreePixmap(XtDisplay(canvas), bitmap);

/* Now free the bitmap */
XFreePixmap(XtDisplay(canvas), pix);

/* Now flush the queue to X-server */
XFlush(XtDisplay(canvas));
}

/*
Routine: xs_clr_win()
Description: Clear the screen
Return: Nothing
Author: Rick Lee
*/
xs_clr_win()
{
    XClearArea(XtDisplay(canvas), XtWindow(canvas), 0, 0, 0, 0, TRUE);
}

/*
Routine: xs_draw_box(x, y, width, height)
Description: Draw a box on screen
Return: Nothing
Author: Rick Lee
*/
xs_draw_box(x, y, width, height)
int x;
int y;
int width;
int height;
{
    XDrawRectangle(XtDisplay(canvas), XtWindow(canvas), line_gc,
                  x, y, width, height);
}

/*
Routine: xs_string
Description: Draw a string on the screen
Return: Nothing
Author: Rick Lee
*/
xs_string(x, y, str)
int x;
int y;
char *str;
{
    XDrawString(XtDisplay(canvas), XtWindow(canvas), line_gc,
               x, y, str, strlen(str));
}

/*
Routine: xs_setcolor
Description: Set the current color
Return: Nothing
Author: Rick Lee
*/
xs_setcolor(color)
int color;
{
    gcw.foreground = color;
    XSetForeground(XtDisplay(canvas), line_gc, color);
}

/*
Routine: xs_win_dim
Description: Get the size of the window
Return: Nothing
Author: Rick Lee
*/
xs_win_dim(width, height)
int *width;
int *height;
{
    Arg args[2];
    Dimension win_width, win_height;

    XtSetArg(args[0], XtNwidth, &win_width);
    XtSetArg(args[1], XtNheight, &win_height);
    XtGetValues(canvas, args, 2);

    *width = win_width;
    *height = win_height;
}

/*
xs_set_cmap(w)
Widget w;
{
    int ncolors;
    XColor colors[256];
    Colormap my_cmap;
    int count = 1;

    Display *d;
    int scr;
    Colormap def_colormap;
    XColor color;
    int xi;
    int the_color = 1000;

    d = XtDisplay(w);
    scr = DefaultScreen(d);
    ncolors = DisplayCells(d, scr);
    def_colormap = DefaultColormap(d, scr);
    for(x = 0; x < ncolors; x++)
    {
        colors[x].pixel = x;
    }
}

```



-419-

[illegible]

【図247】

```

// Filename: zero_vpp.c
// This routine contains the lines to find blocks that have non-zero vpp
// (level 1 segmentation)
//
// =====
//
#include 'charlist.h'
#include 'images.h'
//
// =====
// Scan a line quickly (every 1/4 pixel) looking for an 'on' pixel
int line_scan_quick( // Returns 1 for pixel found, 0 otherwise
    images* line, // The line to process
    int c) // The column to scan
{
    int r;
    for (r=0; r<line->sizeR; r++)
        if (line->pixel(r)(c) & 1)
            return 1;
    return 0;
}

// =====
// Scan a line completely looking for an 'on' pixel
int line_scan_complete( // Returns 1 for pixel found, 0 for otherwise
    images* line, // The line
    int c) // The column to scan
{
    int r;
    for (r=0; r<line->sizeR; r++)
        if (line->pixel(r)(c) & 1)
            return 1;
    return 0;
}

// =====
// The main routine. Call this to make the list
void extract_zero_vpp_list(
    images* line, // The image of the line to segment
    struct NonZeroVppList *list) // A pointer to the list
{
    int c;
    int left, right;
    for (c=0; c<line->sizeC; c++) // Scan the image
    {
        if (line_scan_quick(line, c)) // if found something
        {
            left = right = c; // Look to the left for the start
            while (--c >= 0)
                if (line_scan_complete(line, c) == 0)
                    break;
            left = c+1;
            c = right; // Keep looking to the right
            while (c < line->sizeC)
                if (line_scan_complete(line, c) == 0)
                    break;
            right = c;
            zero_vpp_add(list, left, right); // Add in the new break
        }
    }
}

```

## フロントページの続き

(72)発明者 シン・ヤン ワング  
 アメリカ合衆国 カリフォルニア州  
 92626, コスタメサ ブルマン ストリー  
 ト 3188 キヤノン インフォメーション  
 システムズ インク. 内

(72)発明者 メザードゥ アール. バエズィー  
 アメリカ合衆国 カリフォルニア州  
 92626, コスタメサ ブルマン ストリー  
 ト 3188 キヤノン インフォメーション  
 システムズ インク. 内

(72)発明者 クリストファー エー. シェリック  
 アメリカ合衆国 カリフォルニア州  
 92626, コスタメサ ブルマン ストリー  
 ト 3188 キヤノン インフォメーション  
 システムズ インク. 内